HOKKAIDO UNIVERSITY

Title	Performance Enhancement of Adaptive Network Congestion Control Using Reinforcement Learning
Author(s)	Aoki, Kazuma; Mitsuhashi, Rikima; Iida, Katsuyoshi et al.
Citation	2026 28th International Conference on Advanced Communications Technology (ICACT), 328-334 https://doi.org/10.23919/icact68090.2026.11431550
Issue Date	2026-02-08
Doc URL	https://hdl.handle.net/2115/100417
Rights	© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works
Type	conference paper
File Information	11431550.pdf

Hokkaido University Collection of Scholarly and Academic Papers : HUSCAP

Performance Enhancement of Adaptive Network Congestion Control Using Reinforcement Learning

Kazuma Aoki
Graduate School of Information Science and Technology,
Hokkaido University, Sapporo, Japan.
aoki.kazuma.s1@elms.hokudai.ac.jp

Rikima Mitsuhashi, Katsuyoshi Iida, and Yoshiaki Takai
Information Initiative Center,
Hokkaido University, Sapporo, Japan.
mitsuhashi@tanega.iic.hokudai.ac.jp,
{iida,ytakai}@iic.hokudai.ac.jp

***Abstract*—As the Internet becomes an indispensable part of social infrastructure, network congestion control plays a crucial role in maintaining communication quality. Existing congestion control algorithms, such as NewReno and CUBIC, use predefined rules, making it difficult to respond optimally to changes in network conditions. To address this issue, recent studies have reported the application of dynamic algorithms utilizing reinforcement learning to congestion control. However, most previous studies have not thoroughly evaluated whether their reinforcement learning-based congestion control algorithms can adequately adapt to network fluctuations. Therefore, in this paper, we propose a novel congestion control algorithm called RL-HKD using reinforcement learning and evaluate its communication performance. The evaluation results confirmed that the proposed method improved the bandwidth utilization by up to 1.39% compared to NewReno and demonstrated that it has better throughput convergence and fair coexistence performance than CUBIC.**

***Index Terms*—Network congestion control, machine learning, reinforcement learning, Deep Q-Network (DQN), RL-HKD congestion control algorithm.**

I. Introduction

As the Internet has become an indispensable part of social infrastructure, network congestion control plays a crucial role in maintaining communication quality. Congestion control is one of the functions built into the TCP protocol at the transport layer, which detects situations where the network reaches an overload with excessive data packet traffic and adjusts the data transmission rate. The transmission rate is controlled by the size of the congestion window (CWND) at each TCP packet sender on the network, and its operation is specified by the congestion control algorithms.

Existing congestion control algorithms can be mainly classified into three categories. The first is the loss-based method, such as NewReno [1] and CUBIC [2], which monitors packet loss. The second method is the delay-based approach, which includes Vegas [3] and BBR [4], that monitors packet round-trip time (RTT). The third is the hybrid method, involving Illinois [5] and YeAH [6], which switches between loss-based and delay-based methods. These methods all use static algorithms based on predefined rules, making it challenging to respond optimally to changes in network conditions.

To address the congestion control optimization issue, recent studies have reported the application of dynamic algorithms utilizing reinforcement learning (RL) to congestion control [7].

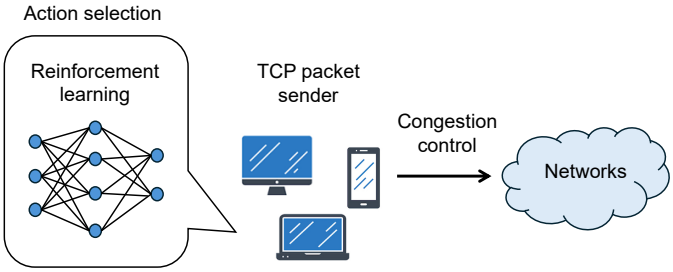


Fig. 1: Concept of the proposed RL-HKD congestion control algorithm using reinforcement learning

However, most previous studies [8], [9] have focused on basic reinforcement learning settings and have not thoroughly evaluated whether their reinforcement learning-based congestion control algorithms can adequately adapt to network fluctuations.

In this paper, we propose a novel congestion control algorithm called RL-HKD using a deep neural network model for reinforcement learning. Fig. 1 shows the concept of the proposed RL-HKD congestion control using reinforcement learning. The TCP packet sender decides the CWND size for congestion control based on action selection using reinforcement learning. The proposed method implements four types of CWND size control in the action space of a deep reinforcement learning model. Additionally, the optimal CWND size at a given point in time is determined based on the rewards derived from throughput and RTT, which indicate the state of the network packet flow.

In the evaluation environment, the proposed congestion control algorithm is implemented on the ns-3 [10] network simulator, and it calls the deep reinforcement learning components on the PyTorch [11] library through the application programming interface (API) provided with ns3-ai [9]. The evaluation results confirmed that the proposed method improved bandwidth utilization by up to 1.39% compared to NewReno and demonstrated better throughput convergence and fair coexistence performance than CUBIC. Due to the recent enormous volume of network traffic on the Internet, a throughput improvement exceeding 1% can have a significant

impact on network performance. As a result, the proposed method provides effective and stable performance compared to existing congestion control algorithms.

The contributions of this paper are summarized as follows:

- We proposed a novel congestion control algorithm using reinforcement learning.
- The proposed method provides effective and stable performance compared to existing congestion control algorithms.
- The proposed method demonstrated the potential for addressing congestion control optimization issues.

The remainder of this paper is organized as follows. Section II summarizes related research on machine-learning-based adaptive congestion control algorithms, including reinforcement learning. Section III explains the proposed method: RL-HKD congestion control algorithm using reinforcement learning. Section IV describes evaluation results based on the single-flow scenario, throughput convergence scenario, and fair coexistence scenario. Section V concludes the paper.

II. Related Work

As the former section introduced, to address the congestion control optimization issue, recent studies have reported the application of dynamic algorithms utilizing reinforcement learning to congestion control [7].

Looking back on history, K. Winstein et al. [12] developed the first machine-learning-based adaptive congestion control algorithm called Remy. The Remy provided a program of congestion control algorithms to run at the endpoints, generated through pre-training in the offline simulations. In their evaluation environment, the algorithm programs generated by Remy demonstrated effective performance compared to human-designed congestion algorithms, including TCP Cubic, Compound, and Vegas. However, Remy had a significant limitation in that it could not adapt to unknown network conditions for which it had not been trained in advance.

In recent years, research has applied dynamic algorithms based on reinforcement learning to traffic congestion control. W. Li et al. [8] proposed an integrated reinforcement-based Q-learning framework with TCP design called QTCP. QTCP enables senders to learn the optimal congestion control policy in an online method. However, QTCP is not a congestion control algorithm based purely on reinforcement learning, as it combines a state reduction technique, known as Kanerva Coding, for training. Therefore, their proposed method does not demonstrate the performance of a congestion control algorithm that is solely composed of reinforcement learning. Additionally, while they introduced three types of CWND size control actions (10-byte increasing, 1-byte decreasing, and letting the CWND remain) for reinforcement learning, there remains significant potential to improve throughput by refining these actions to better adapt to dynamic network conditions.

For another example, H. Yin et al. [9] designed a module that connects the ns-3 network simulator and multiple Python-based AI frameworks, namely ns3-ai. This module is built on a shared memory implementation for an interprocess communication (IPC) mechanism. The evaluation shows that the IPC transfer speed is much faster than that of ns3-gym [13]. The ns3-ai provides a sample code for RL-TCP, which implements a congestion control algorithm using reinforcement learning. The RL-TCP is designed with a reinforcement learning model, action selection, and reward design, but its performance relative to existing congestion control algorithms has not been demonstrated.

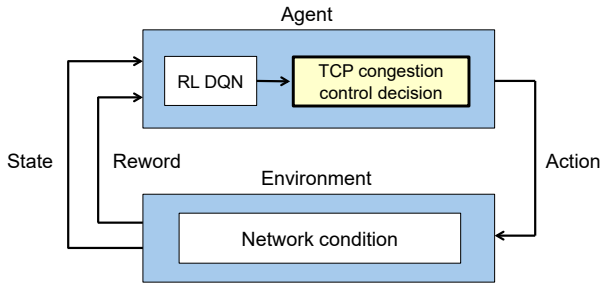


Fig. 2: Deep reinforcement learning model components and flow for TCP congestion control of RL-HKD.

As mentioned above, most previous studies have focused on basic reinforcement learning settings and have not thoroughly evaluated whether their reinforcement learning-based congestion control algorithms can adequately adapt to network fluctuations.

This study explains the design procedure for the proposed congestion control algorithm using reinforcement learning and shows that it provides effective and stable performance compared to existing congestion control algorithms.

III. Proposed method: RL-HKD

As described in the previous section, although most previous studies have applied reinforcement learning to congestion control, their design and evaluation still leave room for further improvement. So, in this section, we propose a comprehensive reinforcement learning-based congestion control algorithm.

The proposed congestion control algorithm, named RL-HKD, is constructed using the environment-agent framework of a Deep Q-Network (DQN) as shown in Fig. 2. The framework contains RL DQN, TCP congestion control decision, and Network condition.

A. Network condition (State)

The network conditions are complex and constantly changing, and network conditions are deeply involved not only in user utilization but also in public and commercial services. In this context of explosive network changes, evaluating network state using every conceivable metric is impractical. Therefore, we focus on reducing the set of state variables for network congestion control. Considering the state space used in RL-TCP [14], we select the five features are used in a state vector: Segments acked, Bytes in flight, Slow start threshold, CWND, and Segment size. “Segments acked” refers to the number of data segments that have been acknowledged upon receiving ACK packets. “Bytes in flight” represents the total amount of data that has been sent but has not yet been acknowledged by

TABLE I: DQN model for reinforcement learning

Model parameter	Value
Input layer	Linear (5 nodes)
Hidden layer	Linear (20 nodes)
Output layer	Linear (4 nodes)
Loss function	MSE
Optimizer	Adam
Learning rate	0.0001

ACK packets. "Slow start threshold" represents the congestion window size at which TCP switches from the slow start phase to the congestion avoidance phase. "CWND" is defined in Eq. 1. Segment size is fixed at 1448 Bytes, which is a standard value used in most networks.

B. TCP congestion control decision (Action)

The selection of Action is the most critical component determining the congestion control performance of RL-HKD. RL-HKD introduces four actions to increase CWND: Each action is defined by Eq. 1. The α takes 1, 2, 3, and 4. When considering the network state at a given point in time, if α takes 1, RL-HKD performs identically to NewReno. On the other hand, if α takes 2, 3, or 4, RL-HKD can provide twice, three times, or four times the capacity compared to NewReno. While NewReno monotonically increased CWND during TCP congestion avoidance, RL-HKD can flexibly select the CWND increment based on network conditions. Regarding the slow start and the reducing CWND size algorithm, RL-HKD uses the same algorithm as NewReno.

$$\text{new_cwnd} = \text{CWND} + \alpha \times \max\left(1, (\text{Segment size})^2/\text{CWND}\right) \tag{1}$$

C. RL DQN (Reward and model)

The utility function defines the objective of RL-TCP. The goal of RL-TCP is to obtain the TCP congestion control decision that maximizes the utility function value for a TCP packet sender. RL-HKD defines utility functions as Eq. 2. The throughput in the equation is calculated as $CWND/RTT$, where RTT is generally measured in milliseconds. The β and γ take 1. If the change in utility ($U = U_{\text{new}} - U_{\text{old}}$) exceeds a predefined threshold ϵ, a reward of +1 is given. Then, if it falls below the threshold, a reward of -1 is given. Based on parameter-tuning results, ϵ is set to -10. RL-HKD can evaluate actions according to changes in network conditions because the rewards are designed based on relative changes in utility.

$$\text{Utility} = \beta \times \text{Throughput} - \gamma \times \text{RTT} \tag{2}$$

As mentioned at the beginning of this section, RL-HKD incorporates the DQN for reinforcement learning, and Table I shows the model structure. The model has three layers: an input layer, a hidden layer, and an output layer. The DQN model accepts the network condition with five features and outputs TCP congestion control decision from four actions. The model is designed by referring to the RL-TCP in ns3-ai [9].

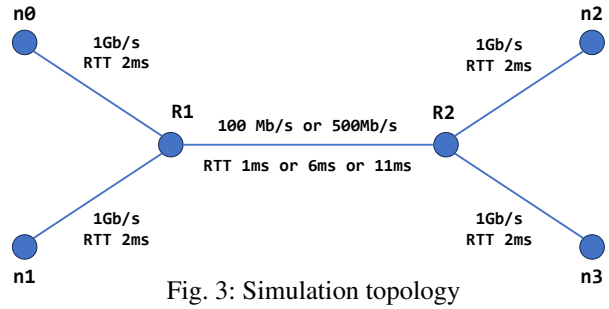


Fig. 3: Simulation topology

IV. Performance Evaluation

The former section presented the design procedure of the proposed network congestion control algorithm using reinforcement learning constructed with the DQN. In this section, we describe the RL-HKD performance evaluation of the proposed method.

A. Evaluation environment setup

The proposed congestion control algorithm is implemented on the ns-3 [10] network simulator, and it calls the deep reinforcement learning components on the PyTorch [11] library through the API provided with ns3-ai [9]. The evaluation environment is built with ns-3 3.38, ns3-ai 1.2.0, Python 3.10.12, Pytorch libtorch-cxx11 on Ubuntu 22.04.4 LTS on two machines with NVidia GeForce RTX3080 and RTX3090.

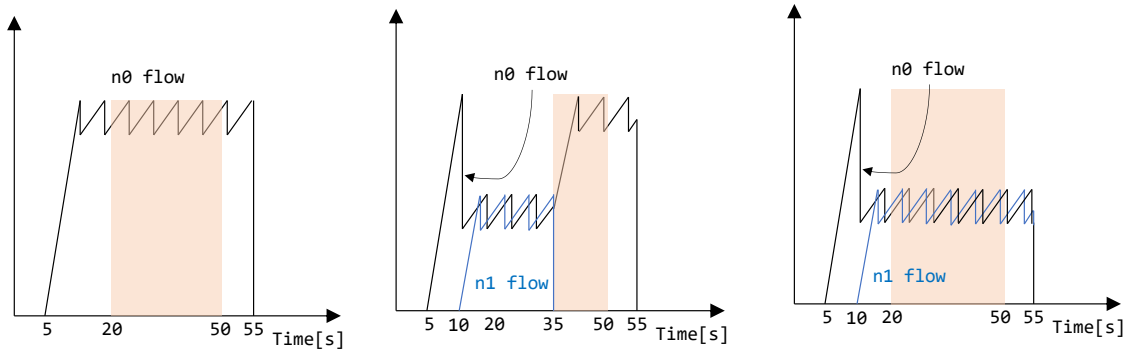
The performance evaluation is based on the dumbbell topology shown in Fig. 3. The n0 and n1 on the left side are clients as TCP packet senders, and the n2 and n3 on the right side are servers as TCP packet receivers. The intermediate R1 and R2 routers form a bottleneck link. In the bottleneck link, the bandwidth (BW) is set to 100 Mb/s and 500 Mb/s, and the RTT is set to 1, 6, and 11 ms. Each client and server forms an access link with the closest router. In the access link, the bandwidth is 1 Gbps and the RTT is 2 ms.

In most of the evaluation scenarios shown in the following subsection, we have two TCP flows (n0 to n2, and n1 to n3), except for the single-flow scenario, which uses only one TCP flow (n0 to n2). In the remainder of this paper, we use simplified terminologies of **n0 flow** and **n1 flow** to present *n0 to n2 flow* and *n1 to n3 flow*, respectively.

B. Evaluation scenario

In order to evaluate the proposed congestion control algorithm from various perspectives, Fig. 4 illustrates the evaluation scenarios.

The first scenario is the single-flow scenario, which is depicted in Fig. 4(a). This is an elementary evaluation of congestion control algorithms, and shows the throughput and bandwidth utilization metrics in the steady-state condition. The following scenario is the throughput convergence scenario in Fig. 4(b). In this scenario, there are n0 and n1 flows. After terminating n1 flow, we evaluate the performance of n0 flow. This scenario is to evaluate the transient performance of congestion control algorithms. The final scenario is the fair



(a) Single-flow scenario (b) Throughput convergence scenario (c) Fair coexistence scenario

Fig. 4: Illustration of evaluation scenarios

TABLE II: Throughput [Mb/s] in single-flow scenario: NewReno v.s. RL-HKD

BW [Mb/s]	100	100	100	500	500	500
RTT [ms]	5	10	15	5	10	15
NewReno	96.39	93.54	89.95	482.01	467.63	443.07
RL-HKD	96.39	94.10	90.58	482.01	467.92	450.02

TABLE III: Utilization [%] in single-flow scenario: NewReno v.s. RL-HKD

BW [Mb/s]	100	100	100	500	500	500
RTT [ms]	5	10	15	5	10	15
NewReno	96.39%	93.54%	89.95%	96.40%	93.52%	88.61%
RL-HKD	96.39%	94.10%	90.58%	96.40%	93.58%	90.00%
Improvement	0.00%	0.56%	0.63%	0.00%	0.06%	**1.39%**

coexistence scenario in Fig. 4(c). In this scenario, we mainly consider the fairness metric under the existence of n0 and n1 flows. Note that the proposed RL-HKD congestion control algorithm uses the DQN, which involves random elements in the reward and state. Therefore, the throughput of RL-HKD in all performance evaluation scenarios represents the average of five trials.

C. Evaluation results

1) Single-flow scenario: The single-flow scenario measures the throughput and bandwidth utilization using NewReno and RL-HKD. This scenario is depicted in Fig. 4(a), the simulation time is set to 55 s, and n0 client begins transmission 5 s after the start of the simulation. The evaluation time window is defined as 20 to 50 s, which is shown in the orange belt in the figure.

Table II displays the results of the average throughput. Comparing the average throughput of NewReno and RL-HKD, RL-HKD achieved equal to or greater than that of NewReno in all network environments.

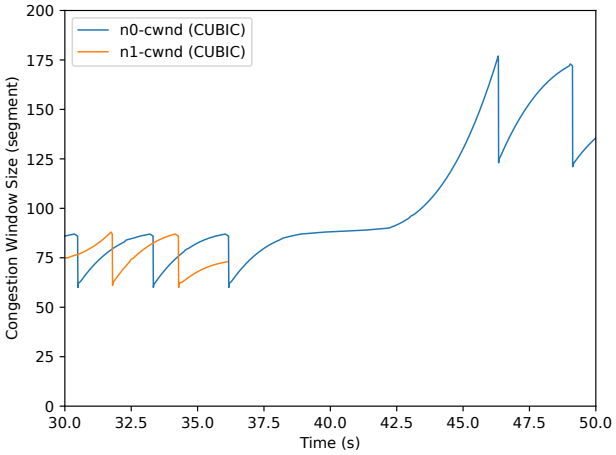
Table III shows improvements of RL-HKD in the bandwidth utilization compared with NewReno. In the network environment with the BW of 500 Mb/s and RTT of 15 ms, the utilization rate of NewReno was 88.61% and that of RL-HKD was 90.00%. As a result, the evaluation result confirmed that the proposed method improved the bandwidth utilization by up to 1.39% compared to NewReno. Given the substantial growth in Internet traffic, a throughput improvement of over 1% can produce a considerable impact on network capability.

2) Throughput convergence: The throughput convergence scenario measures the throughput of RL-HKD and of CUBIC in the transient case. This scenario is depicted in Fig. 4(b), and the simulation time is set to 55 s. The n0 client begins transmission 5 s after the start of the simulation, and the n1 client starts 10 s later. The n1 client terminates its transmission at 35 s. The evaluation time window is defined as 35 to 50 s, which is shown in the orange belt in the figure. This scenario is intended to evaluate the transient performance of the congestion control algorithms used in n0 flow after suddenly finishing n1 flow.

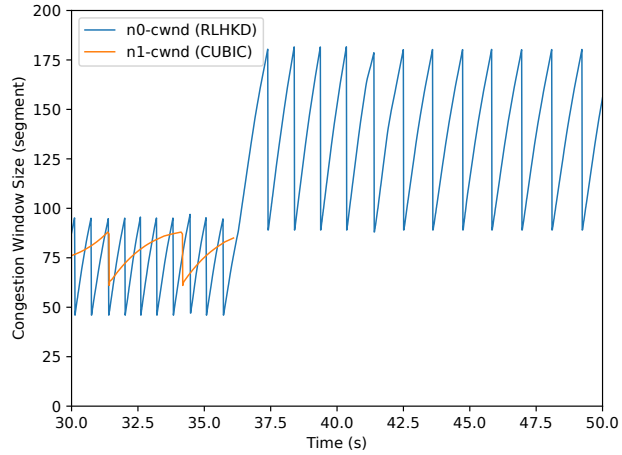
Table IV shows the throughput performance in this scenario. In the table, the first line shows the results under n0 flow with CUBIC and n1 flow with CUBIC, whereas the second line shows the results under n0 flow with RL-HKD and n1 flow with CUBIC. Comparing the throughput of CUBIC and RL-HKD, RL-HKD achieved a greater throughput than CUBIC in all configurations. As a result, the evaluation demonstrated that RL-HKD exhibits superior transient performance compared to CUBIC.

To analyze this further, Fig. 5 illustrates the example plots of CWND traces in CUBIC v.s. CUBIC and in RL-HKD v.s. CUBIC with the bottleneck link bandwidth of 100 Mb/s and RTT of 15 ms, where the relative improvement over CUBIC was the largest and the average throughput was the highest. In both plots, n1 CUBIC flow is terminated at 35 s; consequently, CWND of n0 flow will be increased. In the figure, the CWND increment trend of RL-HKD is greater than CUBIC, which results in greater throughput performance as shown in Table. IV.

3) Fair coexistence: The fair coexistence scenario measures the throughput and Jain's fairness index in NewReno v.s. CUBIC, RL-HKD v.s. CUBIC, and RL-HKD v.s. NewReno cases. This scenario is depicted in Fig. 4(c), and the simulation time is set to 55 s. The n0 client begins transmission 5 s after the start of the simulation, and to obtain stable throughput, the n1 client



(a) CUBIC v.s. CUBIC (Bandwidth of 100 Mb/s, RTT of 15 ms)



(b) RL-HKD v.s. CUBIC (Bandwidth of 100 Mb/s, RTT of 15 ms)

Fig. 5: CWND trace plots in throughput convergence scenario

TABLE IV: Throughput [Mb/s] in convergence scenario

BW [Mb/s] RTT [ms]	100 5	100 10	100 15	500 5	500 10	500 15
CUBIC (v.s. CUBIC)	90.81	88.38	75.56	473.37	401.67	368.47
RL-HKD (v.s. CUBIC)	**91.45**	**88.80**	**85.62**	**474.45**	**447.60**	**387.78**

TABLE V: Throughput [Mb/s] in fair coexistence scenario

BW [Mb/s] RTT [ms]	100 5	100 10	100 15	500 5	500 10	500 15
NewReno CUBIC	48.19 48.21	46.43 49.93	44.73 50.81	240.99 241.02	232.48 249.45	223.61 258.26
NewReno RL-HKD	48.19 48.21	47.66 48.64	46.45 49.16	240.99 241.02	239.11 240.07	232.61 240.40
CUBIC RL-HKD	48.20 48.19	49.45 46.92	50.46 45.16	241.01 241.01	249.12 232.63	253.08 224.63

starts 10 s later. The evaluation time window is defined in 20 to 50 s, which is shown in the orange belt in the figure.

The Jain's fairness index [15] is defined as

$$\frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \cdot \sum_{i=1}^{n} x_i^2},$$

where n is the number of flows, and x_i is the obtained throughput of i-th flow. If the throughput is completely fair shared, then the index becomes 1.0.

Table V shows the throughput in NewReno v.s. CUBIC, NewReno v.s. RL-HKD, and CUBIC v.s. RL-HKD cases. When comparing the average throughput of RL-HKD and NewReno, RL-HKD achieves higher throughput than NewReno in all configurations.

Table VI demonstrates RL-HKD fair coexistence on Jain's fairness index. The table presents index values rounded to the fifth decimal place. An index value of exactly 1.0000 is marked with an asterisk (*) to denote that all flows are distributed with complete fairness. If a value of 1.0000 appears without the asterisk, this indicates that the flows approximate complete fairness but do not achieve it precisely.

NewReno v.s. CUBIC flows recorded a minimum value of 0.9949, while RL-HKD v.s CUBIC flows attained a minimum value of 0.9965. These results indicate that RL-HKD achieves higher fairness than NewReno in a coexistence environment with CUBIC.

Moreover, although NewReno v.s. CUBIC flows obtained a minimum value of 0.9949, RL-HKD v.s. NewReno achieved a minimum value of 0.9992. These results suggest that RL-HKD achieves higher fairness than CUBIC in a coexistence environment with NewReno. In other words, the evaluation results confirmed that the proposed method demonstrated better fair coexistence performance than CUBIC.

4) Comparison with previous studies: In order to compare the performance with previous studies that used reinforcement learning for congestion control, the scenario measures the performance of single-flow average throughput using the RL-HKD and RL-TCP. The RL-TCP is publicly available sample code that implements a congestion control algorithm using reinforcement learning in ns3-ai [9]. The RL-TCP has a reinforcement-based Q-learning and DQN framework for TCP congestion control [14]. To compare under conditions as similar as possible to the proposed method, this experiment uses the DQN framework in RL-TCP.

The simulation timeline is precisely the same as the single-flow scenario in Fig. 4(a).

Table VII displays the throughput results of RL-TCP and of RL-HKD. Comparing the average throughput of RL-TCP and RL-HKD, RL-HKD achieves equal to or greater than that of RL-TCP in most network environments, except for the condition that the bottleneck link bandwidth is 500 Mb/s and the RTT is 15 ms.

Table VIII shows the bandwidth utilization of RL-TCP and of RL-HKD. In the condition with BW of 100 Mb/s and RTT of 15 ms, the utilization of RL-TCP was 89.70% and that of RL-

TABLE VI: Jain's fairness index in fair coexistence scenario

BW [Mb/s]	100	100	100	500	500	500
RTT [ms]	5	10	15	5	10	15
NewReno v.s. CUBIC	1.0000	0.9987	0.9960	1.0000	0.9988	0.9949
RL-HKD v.s. CUBIC	**1.0000**	**0.9993**	**0.9969**	**1.0000 (*)**	**0.9988**	**0.9965**
RL-HKD v.s. NewReno	**1.0000**	**0.9999**	**0.9992**	**1.0000**	**1.0000**	**0.9997**

HKD was 90.58%. As a result, the evaluation result confirmed that the proposed method improved the bandwidth utilization by up to 0.88% compared to RL-TCP. As a result, a throughput improvement exceeding 0.8% has been observed compared to prior research. This result indicates that the proposed method contributes to the advancement of network congestion control and has a significant impact on overall performance.

One should note that RL-HKD fell short of RL-TCP by 0.56% under the conditions of the bottleneck link bandwidth of 500 Mb/s and the RTT of 15 ms. Since the finding may appear unfavorable, we have to analyze its reasons and enhance the proposed method, which is another future research topic.

TABLE VII: Throughput [Mb/s]: RL-TCP v.s. RL-HKD

BW [Mb/s]	100	100	100	500	500	500
RTT [ms]	5	10	15	5	10	15
RL-TCP	96.39	93.62	89.70	482.01	466.74	452.78
RL-HKD	96.39	94.10	90.58	482.01	467.92	450.02

TABLE VIII: Utilization [%]: RL-TCP v.s. RL-HKD

BW [Mb/s]	100	100	100	500	500	500
RTT [ms]	5	10	15	5	10	15
RL-TCP	96.39%	93.62%	89.70%	96.40%	93.35%	90.56%
RL-HKD	96.39%	94.10%	90.58%	96.40%	93.58%	90.00%
Improvement	0.00%	0.48%	**0.88%**	0.00%	0.23%	-0.56%

D. Discussion

As demonstrated above, the proposed method provides effective and stable performance compared to existing congestion control algorithms. The performance enhancement is due to the contribution of reinforcement learning. Its strength lies in its ability to select flexible and optimal actions in response to changes in network conditions.

On the other hand, reinforcement learning requires high computing resources, including GPUs. Currently, it is being executed in a simulation environment, but in a real environment, insufficient computing resources for reinforcement learning may lead to communication delays. In the simulation environment, ns-3 stops the simulation time during DQN execution. Although this problem may be solved with the advancement of GPUs, performance evaluation in a real environment remains a future challenge.

This study primarily focused on designing the action space and the reward function. Therefore, improving the neural network model remains an important direction for future work. In addition, since the evaluation was conducted in simple network environments, we also plan to assess the performance in environments with frequent changes, such as wireless communication networks.

V. Conclusion

Existing congestion control algorithms, such as NewReno and CUBIC, use predefined rules, making it difficult to respond optimally to changes in network conditions. To address this issue, we propose the RL-HKD congestion control algorithm, which utilizes reinforcement learning. The proposed method provided effective and stable performance compared to existing congestion control algorithms, demonstrating its potential in addressing congestion control optimization issues. Future work includes applying the proposed congestion control algorithm in a real environment.

References

[1] A. Gurtov, T. Henderson, S. Floyd, and Y. Nishida, "The NewReno modification to TCP's fast recovery algorithm," IETF RFC 6582, Apr. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6582

[2] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, July 2008.

[3] L. Brakmo and L. Peterson, "TCP vegas: end to end congestion avoidance on a global internet," *IEEE J. Selected Areas in Commun.*, vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

[4] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time," *ACM Queue*, vol. 14, no. 5, pp. 20–53, Oct. 2016.

[5] S. Liu, T. Başar, and R. Srikant, "TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks," *Performance Evaluation*, vol. 65, no. 6, pp. 417–440, June 2008.

[6] A. Baiocchi, A. P. Castellani, and F. Vacirca, "YeAH-TCP: Yet another highspeed tcp," in *Proc. Int'l Workshop on Protocols for FAST Long-Distance Networks (PFLDnet07)*, vol. 7, Feb. 2017, pp. 37–42.

[7] T. Zhang and S. Mao, "Machine learning for end-to-end congestion control," *IEEE Commun. Mag.*, vol. 58, no. 6, pp. 52–57, June 2020.

[8] W. Li, F. Zhou, K. R. Chowdhury, and W. Meleis, "QTCP: Adaptive congestion control with reinforcement learning," *IEEE Trans. Network Science and Engineering*, vol. 6, no. 3, pp. 445–458, July-Sept. 2019.

[9] H. Yin, P. Liu, K. Liu, L. Cao, L. Zhang, Y. Gao, and X. Hei, "ns3-ai: Fostering artificial intelligence algorithms for networking research," in *Proc. ACM 2020 Workshop on ns-3*, June 2020, pp. 57–64.

[10] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Güneş, and J. Gross, Eds. Berlin, Heidelberg: Springer, Feb. 2010, pp. 15–34.

[11] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," Dec. 2019, arXiv preprint arXiv:1912.01703.

[12] K. Winstein and H. Balakrishnan, "TCP ex machina: computer-generated congestion control," in *Proc. ACM SIGCOMM 2013*, Oct. 2013, pp. 123–134.

[13] P. Gaw lowicz and A. Zubow, "ns-3 meets OpenAI gym: The playground for machine learning in networking research," in *Proc. ACM Int'l Conf. Modeling, Analysis and Simulation of Wireless and Mobile Systems (MWIM'19)*, Nov. 2019, pp. 113–120.

[14] DianGroup, "RL-TCP example," Mar. 2024. [Online]. Available: https://github.com/hust-diangroup/ns3-ai/tree/main/examples/rl-tcp

[15] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *DEC Research Report*, vol. TR-301, Sept. 1998.

Kazuma Aoki received the B.E. degree in Computer Science and Information Technology from Hokkaido University in 2024. Currently, he is a master's course student at the Graduate School of Information Science and Technology, Hokkaido University. His research interests are congestion control algorithms using machine learning on the Internet.

Rikima Mitsuhashi (Member, IEEE) received Bachelor of Informatics in Computer Science from Shizuoka University in 2000, M.E. in Computational Intelligence and Systems Science from Tokyo Institute of Technology in 2002, and Ph.D. in Information Science and Technology from the University of Tokyo in 2022. He is currently a Visiting Researcher for the Information Initiative Center, Hokkaido University. His research interests include secure computing, network security, and artificial intelligence. He is a member of IEICE, and IPSJ.

Katsuyoshi Iida (Member, IEEE) received the B.E., M.E., and Ph.D. degrees in Computer Science and Systems Engineering from Kyushu Institute of Technology (KIT), Iizuka, Japan in 1996, in Information Science from Nara Institute of Science and Technology, Ikoma, Japan in 1998, and in Computer Science and Systems Engineering from KIT in 2001, respectively. Currently, he is a Professor at the Information Initiative Center, Hokkaido University, Sapporo, Japan. His research interests include network systems engineering such as network architecture, QoS, and cyber-security. He is a fellow of IEICE, Japan. He received the 18th TELECOM System Technology Award, and Tokyo Tech young researcher's Award in 2003, and 2010, respectively.

Yoshiaki Takai (Member, IEEE) received the B.E. degree in electronic engineering and the M.E. and Ph.D. degrees in information engineering from Tohoku University in 1983, 1985, and 1988, respectively. From 1988 to 1989, he was a Research Associate with the Faculty of Science, The University of Tokyo. In 1989, he joined the Faculty of Engineering, Hokkaido University. From 2011 to 2019, he was a Director of the Information Initiative Center, Hokkaido University. He is currently a Professor with the Information Initiative Center, Hokkaido University. He is also an Executive Adviser for CIO of Hokkaido University. His research interests include parallel computer architecture, parallel processing, distributed processing, computer networks, augmented reality, virtual reality, physically based modeling, and computer graphics applications. He is a member of IEICE and IPSJ.