ACKurate-DQN: Adaptive Throughput-Based Congestion Control Using Deep Reinforcement Learning

Heeju Chae*, Wanseon Lim†, Gosan Noh‡ and Eunkyung Kim*
*Dept. of Artificial Intelligence, Hanbat National University, Sejong, Korea
heejuchae@edu.hanbat.ac.kr, ekim@hanbat.ac.kr
†Fiscal and Economic Policy Intelligence Research Center,
Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea
wslim@etri.re.kr
‡Dept. of Electronic Engineering, Hanbat National University, Daejeon, Korea
gsnoh@hanbat.ac.kr

***Abstract*—ACKurate-DQN is a *Deep Q-Network (DQN)* TCP controller that replaces hand-tuned *Additive Increase, Multiplicative Decrease (AIMD)* logic with a single, scale-invariant reward. A lightweight dual-mode estimator tracks the path's *throughput ceiling*, and the agent automatically balances throughput and delay to maximize performance, without manual weight tuning. In ns-3 dumbbell-topology tests, ACKurate-DQN sustains more than 95% link utilization and RTT comparable to a recent DQN-AIMD baseline across both static links and a bandwidth-shift scenario with a fixed RTT. This ceiling-aware feedback stabilizes learning and enables more robust operation than existing DQN-based AIMD methods.**

***Index Terms*—TCP congestion control, reinforcement learning, deep Q-network, adaptive throughput estimation**

I. Introduction

Classical Additive Increase, Multiplicative Decrease (AIMD) controllers [3]–[5] break down when bandwidth or queuing delay shifts within a few Round-Trip Times (RTTs). Recent reinforcement learning (RL) approaches adjust the congestion window proactively, yet still rely on hand-tuned reward weights, as seen in Aurora [6], TCP-Drinc [7], and ORC [8]. Instead, Seo and Cho [1] normalize reward with a self-adjusting throughput ceiling $T_{\max}$ so that the reward remains effective across diverse bandwidth and RTT conditions. We further refine this ceiling to make the reward smoother and more responsive.

ACKurate-DQN consistently achieves higher link utilization while maintaining RTTs comparable to the baseline across diverse network conditions, outperforming the DQN-based AIMD baseline [1], [2]. It reuses the neural architecture proposed by Seo and Cho [1], and adopts the broader action set introduced in their follow-up work [2]. To the best of our knowledge, the method proposed by Seo and Cho [1] remains the most effective RL-based congestion control algorithm implemented in `ns-3`, which is also the simulation environment used in our experiments. The key contributions of this work are summarized below.

Adaptive reward normalization. We propose a lightweight dual-mode estimator that infers bandwidth capacity from both the Exponentially Weighted Moving Average (EWMA) throughput and the instantaneous throughput, as detailed in Section III (see Fig. 1). This estimator adjusts the throughput ceiling $T_{\max}$ for normalizing the current throughput T_{curr}, which is calculated from ACKed segments during the action interval, yielding a scale-invariant reward without manual weight tuning:

$$r = \frac{T_{\text{curr}}}{T_{\max}}.$$

This formulation helps stabilize learning across diverse network conditions.

Comprehensive performance comparison. We evaluate ACKurate-DQN against a DQN-AIMD baseline [1] across a wide range of static and dynamic bottlenecks, observing consistently higher throughput and similar RTT performance.

The rest of the paper is structured as follows: Section II reviews relevant literature; Section III describes the architecture underlying ACKurate-DQN; Section IV presents our experimental setup, evaluation methodology, and comprehensive results and baseline comparisons. Finally, Section V concludes the paper and outlines potential avenues for future research.

II. Related Work

AIMD legacy. Reno, NewReno, and CUBIC increase `cwnd` linearly and halve it upon loss, which can lead to link under-utilization or queue buildup when capacity fluctuates rapidly [3], [4], [5].

RL-augmented AIMD. DQN-based AIMD methods [1], [2] retain AIMD fallbacks for safety: a loss event resets `cwnd` to a predefined value, ignoring the network state inferred by the agent. This delegates the initial recovery step to AIMD, while the RL agent remains responsible for congestion prevention and subsequent adjustments.

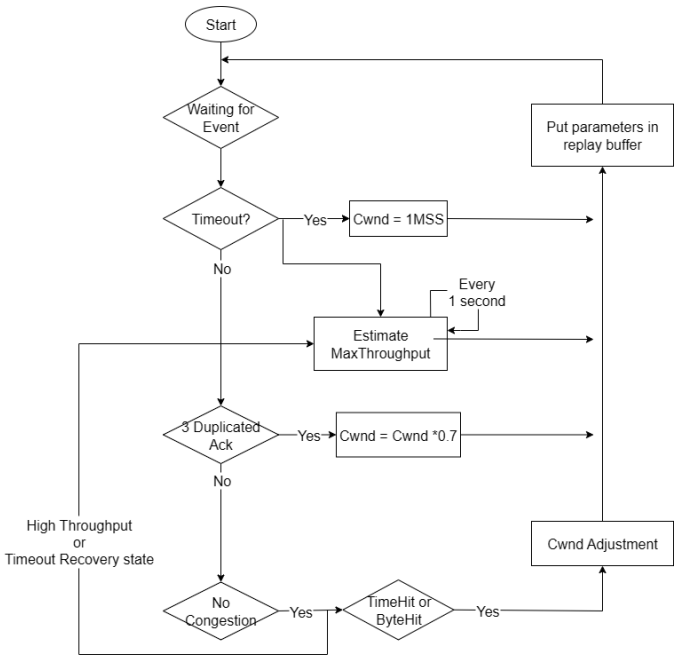


Fig. 1: Overall ACKurate-DQN decision process. Timeout events trigger conservative cwnd reset, while cwnd is otherwise adjusted via duplicate ACKs or pacing-based policy updates. Throughput estimation is handled separately and used for updating $T_{\max}$.

Pure-RL controllers. Aurora [6] learns the sending rate directly from delay/throughput traces; TCP-Drinc (DQN) [7] adjusts `cwnd` using discrete Q-learning actions; ORC (actor-critic) [8] accelerates online adaptation using a pre-trained actor–critic that refines itself in real time. All three eliminate AIMD fallbacks but still require carefully tuned reward functions or training curricula to generalize across network conditions.

III. ACKurate-DQN Design

A. ACKurate-DQN Decision Workflow and Control Logic

Fig. 1 illustrates the complete decision-making workflow of ACKurate-DQN. The agent continuously monitors network feedback and selectively adjusts the congestion window to respond to both performance dynamics and congestion signals.

a) Throughput Ceiling ($T_{\max}$): Every second, the agent re-evaluates $T_{\max}$. If the recent EWMA throughput falls below $T_{\max}$, the ceiling is conservatively reduced. If either the EWMA throughput or the instantaneous throughput approaches $T_{\max}$, or the sender is in timeout recovery, the ceiling is gently increased. This minimalist rule keeps most rewards $r = T_{\text{curr}}/T_{\max}$ safely within $[0, 1]$, without requiring explicit RTT or BDP information.

b) Event-Driven Congestion Window: A retransmission timeout forces `cwnd`=1 MSS, while three duplicate ACKs reduces it to 0.7 `cwnd`. Otherwise, whenever the newly acknowledged bytes exceed 1.5 `cwnd` *or* the elapsed time surpasses 1.5 RTTs, the RL policy selects the next `cwnd`. Thus, classical safety guards are complemented by learned, feedback-aware adjustments in a few concise lines of code.

B. State, Action, and Reward

State. Each decision uses six scalar inputs: current `cwnd`, instantaneous throughput, ACK count, interval Δt, and Aurora-style latency gradient and ratio metrics.

Action. The DQN chooses an additive window change from the set $\{-10, -3, -1, 0, 1, 3, 10\}\times$MSS, allowing both cautious and aggressive probes.

Reward. A single scale-free metric $r = T_{\text{curr}}/T_{\max} \in [0, 1]$ encourages high utilization while implicitly penalising delay because queue build-up lowers current throughput.

IV. Experiments

A. Experimental Setup

Fig. 2: Dumbbell network topology used in our experiments.

All experiments are conducted using `ns-3.36`, interfaced via `ns3-gym`. The simulated network uses a classical dumbbell topology (Fig. 2), in which a TCP sender communicates with a receiver through two intermediate routers. The leftmost node acts as the *Sender*, connected to *Router A*. This router forwards packets over a time-varying bottleneck link to *Router B*, which then delivers them to the rightmost node, the *Receiver*. Each TCP packet carries a 1460-byte payload. The bottleneck router uses a DropTail queue with a buffer size equal to 1.5 times the bandwidth–delay product (BDP), allowing controlled queuing under congestion. The RTT is fixed at 100 ms for all scenarios.

Two experiments are conducted to evaluate robustness and adaptability:

- In the first experiment, the bottleneck bandwidth alternates between 25 Mbps and 50 Mbps every 100 seconds, over a total duration of 300 seconds. This setup evaluates the agent's robustness to abrupt bandwidth variations.
- In the second experiment, the agent is tested under five static bandwidth configurations (5, 25, 50, 75, and 100 Mbps), each lasting 300 seconds. This experiment assesses adaptability across diverse bandwidth conditions.

In both experiments, the proposed ACKurate-DQN agent is compared against baseline algorithms. The primary baseline is a DQN-based congestion control method [2], denoted as *DQN-based TCP* throughout this paper. It shares the same neural architecture as ACKurate-DQN, but computes the maximum throughput based on the current congestion window (`cwnd`) and RTT. In contrast, ACKurate-DQN derives throughput directly from ACK feedback, enabling more responsive adaptation to network dynamics.

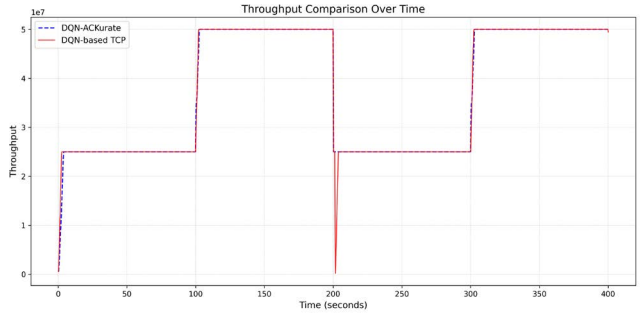


Fig. 3: Throughput over time in a dynamic environment where the bottleneck bandwidth alternates between 25 Mbps and 50 Mbps every 100 seconds. ACKurate-DQN maintains stable throughput, whereas DQN-based TCP shows a significant dip around 200 seconds.

B. Results

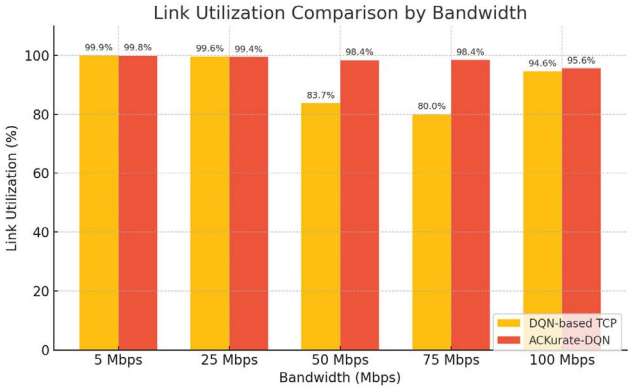
To evaluate the effectiveness of ACKurate-DQN under dynamic and diverse network conditions, we compare its performance against a baseline DQN-based TCP across both time-varying and static bandwidth scenarios.

Dynamic scenario. Fig. 3 depicts the throughput over time when the bottleneck bandwidth alternates between 25 Mbps and 50 Mbps every 100 seconds. Both algorithms generally adapt well to bandwidth shifts, maintaining high throughput. However, DQN-based TCP exhibits a sharp drop around the 200-second mark, failing to promptly readjust after a bandwidth decrease. In contrast, ACKurate-DQN maintains stable throughput, quickly adapting to both increases and decreases in capacity. This highlights its robustness under abrupt network changes.

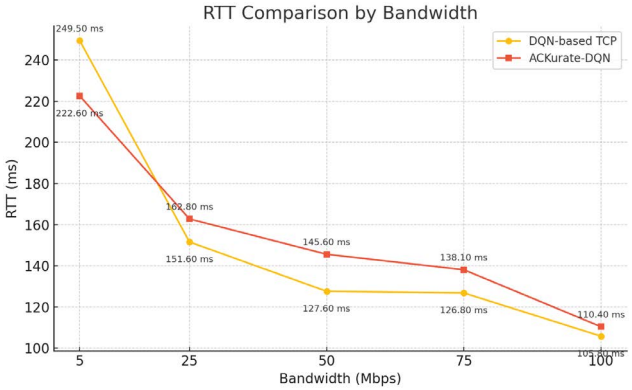
Static scenario. Fig. 4 presents results under static bandwidth conditions. ACKurate-DQN sustains link utilization at or above 95% across all capacities, while the baseline dips to 80% at 75 Mbps. Although ACKurate-DQN shows a modest increase in average RTT (Fig. 4b), its throughput gain more than makes up for the delay. Overall, throughput ceiling feedback enables higher utilization without compromising efficiency.

Together, these results demonstrate that ACKurate-DQN exhibits:

- **Robustness**: It handles sudden bandwidth changes more robustly than the baseline.
- **Adaptability**: It generalizes well across a wide range of network capacities, avoiding both under- and over-utilization.
- **Efficiency**: It achieves high link utilization while maintaining acceptable RTT, balancing throughput and latency effectively.



(a) Link utilization across static bandwidth settings (5–100 Mbps). ACKurate-DQN consistently achieves higher utilization, especially at 50 and 75 Mbps.



(b) RTT across static bandwidth settings. ACKurate-DQN shows lower RTT at 5 Mbps and remains competitive at higher bandwidths.

Fig. 4: Performance comparison between DQN-based TCP and ACKurate-DQN under static bandwidth configurations.

V. Conclusion and Future Work

We propose ACKurate-DQN, a reinforcement learning–based TCP congestion control algorithm that adjusts the congestion window using a dual-mode adaptive throughput ceiling ($T_{\max}$). ACKurate-DQN delivers robust, adaptive performance under diverse network conditions without relying on explicit BDP or RTT estimates. By leveraging lightweight throughput normalization and event-driven feedback, it remains responsive across varying link conditions.

In conclusion, ACKurate-DQN demonstrates robust adaptation to sudden bandwidth changes, generalizes well across static and dynamic environments, and maintains high throughput with low latency. These results highlight its potential as a scalable and practical learning-based congestion control approach that operates effectively without per-topology tuning. While effective in single-flow settings, ACKurate-DQN has not yet been evaluated in multi-flow scenarios or under cross-traffic. Future work includes extending the framework to heterogeneous traffic types and deploying it in real-world platforms using eBPF or kernel modules. ACKurate-DQN represents a practical step toward scalable, learning-driven congestion control that generalizes across networks without environment-specific tuning.

ACKNOWLEDGMENT

This work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) - ITRC (Information Technology Research Center) grant funded by the Korea government (MSIT) (IITP-2025-RS-2024-00437886, 50%) and by Basic Science Research Program through the National Research Foundation of Korea (NRF) grant funded by the Ministry of Education (RS-2025-25408839, 50%).

REFERENCES

[1] S.J. Seo and Y.Z. Cho, "Fairness Enhancement of TCP Congestion Control Using Reinforcement Learning," in *Proc. ICAIIC*, 2022, pp. 288–291. doi: 10.1109/ICAIIC54071.2022.9722626.

[2] S.J. Seo and Y.Z. Cho, "Inter-Protocol Fairness Evaluation of DQN-based Congestion Control Algorithms," in *Proc. ICAIIC*, 2023, pp. 693–696. doi: 10.1109/ICAIIC57133.2023.10067107.

[3] M. Allman, V. Paxson, and W. R. Stevens, "TCP Congestion Control," RFC 5681, RFC Editor, 2009. [Online]. Available: https://www.rfc-editor.org/info/rfc5681

[4] S. Floyd and T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 6582, RFC Editor, 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6582

[5] I. Rhee, L. Xu, S. Ha, and A. Zimmermann, "CUBIC for Fast Long-Distance Networks," RFC 9438, RFC Editor, 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9438

[6] N. Jay, N. H. Rotman, P. B. Godfrey, M. Schapira, and A. Tamar, "A Deep Reinforcement Learning Perspective on Internet Congestion Control," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, Long Beach, CA, USA, 2019, pp. 3050–3059. [Online]. Available: https://proceedings.mlr.press/v97/jay19a.html

[7] K. Xiao, S. Mao, and J. K. Tugnait, "TCP-Drinc: Smart Congestion Control Based on Deep Reinforcement Learning," *IEEE Access*, vol. 7, pp. 11892–11908, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2892046

[8] Y. Li, J. Huang, C. Wu, X. Zhu, and J. Wang, "ORC: Online Reinforcement Learning for Congestion Control with Fast Convergence," in *Proceedings of the ACM Asia-Pacific Workshop on Networking (APNet)*, Hangzhou, China, 2025, pp. 1–7. [Online]. Available: https://dl.acm.org/doi/10.1145/3735358.3735381

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXiv preprint arXiv:1312.5602*, 2013. [Online]. Available: https://arxiv.org/abs/1312.5602