engineering proceedings

MDPI

Proceeding Paper

Proposed Multi-Strategy AQM Framework Based on Cross-Layer Multi-Agent Reinforcement Learning †

Vuong Xuan Chi [1,*], Nguyen Kim Quoc [1] and Ho Khoi [2]

[1] Faculty of Information Technology, Nguyen Tat Thanh University, Ho Chi Minh City 700000, Vietnam; nkquoc@ntt.edu.vn
[2] Institute of Information Technology, Vietnam Academy of Science and Technology, Hanoi 100000, Vietnam; khoih@ioit.ac.vn
* Correspondence: vxchi@ntt.edu.vn
† Presented at the 2026 International Conference on Energy, Infrastructure and Environmental Research (EIER 2026), Ho Chi Minh City, Vietnam, 15–17 July 2026.

Abstract

Network congestion and bufferbloat remain serious issues in modern network systems, especially in AI clusters, edge computing, and IoT environments with high and highly dynamic traffic. Traditional AQM algorithms such as RED, CoDel, and PIE, as well as some models that combine machine learning or deep learning with standalone AQM approaches, still face limitations due to their reliance on static parameters and lack of predictive capability. The paper proposes an AQM framework that integrates the strengths of these algorithms within a machine learning model, using cross-layer reinforcement learning in a multi-agent (MARL) approach. The solution enables coordination between routers and TCP end-hosts to predict congestion and dynamically adjust drop/mark probabilities in real time. Experimental results show improvements in delay, packet loss, jitter, throughput, and fairness.

Keywords: hybrid AQM; AQM framework; multi-agent RL; bufferbloat mitigation; congestion control

1. Introduction

In recent years, the rapid growth of delay-sensitive applications such as large-scale AI training, real-time inference, 4K/8K video streaming, and IoT in edge computing environments exposes the limitations of traditional congestion control mechanisms. The bufferbloat phenomenon-excessive queuing delay caused by oversized or persistently occupied buffers-leads to high end-to-end delay, degrades Quality of Experience (QoE), and significantly impacts the performance of both data center and edge networks [1]. Classical Active Queue Management (AQM) algorithms, including Random Early Detection (RED) [2], Controlled Delay (CoDel) [3], and Proportional Integral controller Enhanced (PIE) [4], aim to mitigate bufferbloat by proactively dropping or marking packets. However, these approaches rely heavily on static parameters (e.g., thresholds and target delay) and lack adaptability to highly dynamic and rapidly varying traffic conditions in modern networks. As a result, their performance remains unstable under bursty traffic and elephant flows commonly observed in AI clusters.

To address these limitations, recent studies have shifted toward the application of ML, particularly Deep Reinforcement Learning (DRL), to AQM [5]. These approaches enable routers to learn from real-time network conditions and adapt congestion control without

Check for updates

Academic Editor: Le Kim Hung

Published: 12 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

manual tuning. Hybrid RL-based models further combine multiple AQM schemes and cross-layer information to improve end-to-end congestion management [6].

The main contribution of this study is the proposal of a novel AQM framework that simultaneously integrates the representative mechanisms of RED, CoDel, and PIE within a cross-layer Multi-Agent Reinforcement Learning model. The proposed framework aims to enhance congestion prediction and enable more effective end-to-end congestion control, particularly in AI clusters, edge computing, and IoT environments.

2. Related Work

2.1. Traditional and Hybrid AQM Algorithms

Classical AQM algorithms such as RED employ probabilistic packet dropping based on the average queue length, CoDel regulates sojourn time to mitigate bufferbloat, and PIE utilizes a proportional–integral (PI) controller to dynamically adjust drop probability [2–4]. Hybrid approaches, including FQ-CoDel and FQ-PIE, improve fairness and flow isolation [7]; however, they lack automatic adaptability to heterogeneous traffic in AI clusters and edge networks.

2.2. Advanced Hybrid and Variant AQM Algorithms

RL-based AQM schemes, such as DQN and PPO, reduce delay and packet loss without manual tuning. For example, DESiRED combines INT and Deep RL to dynamically adjust target delay, outperforming traditional iRED in data center networks [8]. Beyond RED, CoDel, and PIE, numerous variants and hybrid schemes enhance fairness, reduce tuning overhead, and support QoS differentiation in modern environments such as home gateways, IoT, and programmable data planes. Hybrid designs combining Fair Queuing (FQ) with AQM gain prominence from 2010 to 2025, addressing flow isolation and reducing collateral impact on interactive traffic (e.g., VoIP, web browsing). These approaches improve QoE for applications such as DASH streaming, where PIE controls queue delay and FQ-PIE ensures bandwidth fairness and reduces inter-flow contention [9].

Probabilistic variants focusing on fairness or gentle dropping include CHOKe [10] and enhanced BLUE-based approaches (EBA) to reduce congestion and improve transmission efficiency [11]. PAQMAN employs a Semi-Markov Decision Process (SMDP) to derive optimal drop decisions based on TCP probabilistic models and target delay [12]. RED-SP-CoDel integrates RED with Static Priority scheduling and CoDel on programmable data planes (P4), enabling effective QoS differentiation and reduced delay for high-priority flows [13]. Additionally, Delta AQM applies machine learning (mixture density networks) to predict Delay Violation Probability (DVP) for each packet based on end-to-end delay targets [14]. These advances mark a shift from heuristic AQM to hybrid, control-theoretic, and ML-driven solutions for adaptive congestion control in 5G/6G and AI-driven networks.

More recent studies extend RL-based AQM toward multi-agent and IoT-oriented designs. Kim and Jaseemuddin propose a DQN-based AQM with a delay–throughput trade-off reward for fog/edge nodes, outperforming conventional schemes under varying RTT and bandwidth [15]. A weighted ensemble combining DQN, PPO, DDPG, and TD3 further improves robustness of DRL-based congestion control over single-agent designs [16]. LFQ introduces per-flow queue-size learning via deep RL to enhance fairness among heterogeneous flows [17], while MACC explores cross-layer multi-agent RL cooperation between AQM and TCP congestion control [18]. Collectively, these works motivate cross-layer, multi-agent RL as a natural extension of single-agent AQM.

https://doi.org/10.3390/engproc2026164011

3. Research Methods

3.1. Overview of the Proposed Framework

In AQM research for mitigating congestion and bufferbloat at routers, integrating RED, PIE, and CoDel in a hybrid framework is well justified. No single algorithm performs optimally under all conditions, so combining them leverages complementary strengths and mitigates weaknesses. This is important in dynamic environments like AI clusters, edge computing, and SDN, where traffic varies widely. A hybrid design offers balanced performance, improved stability, and better handling of congestion and hetero-geneous flows. Comparative studies show that CoDel and PIE generally achieve lower delay than RED, whereas RED performs better under bursty traffic. PIE is also more stable than CoDel under heavy load. A hybrid framework can reduce queuing delay while maintaining high utilization. In addition, it can be integrated with ML/RL techniques to dynamically adjust weights, prioritizing CoDel for low delay, RED for burst handling, and PIE for stability (Table 1).

Table 1. Comparison of AQM mechanisms: RED, CoDel, and PIE.

Algorithm	Core Mechanism	Advantages	Limitations	Best Use Cases
RED [2]	Probabilistic drop via average queue (EWMA).	Early detection, avoids TCP sync, handles bursty traffic.	Many parameters, difficult to tune, no explicit delay control.	Bursty traffic, traditional networks.
CoDel [3]	Drop based on sojourn time exceeding target	Minimal tuning, reduces bufferbloat, low delay	Round-Trip Time (RTT) sensitivity, less effective under heavy load	Low-delay and real-time networks
PIE [1]	PI controller adjusts drop probability based on delay.	Stable, self-tuning, performs well under high load	Potential jitter, indirect delay estimation.	High-load networks requiring balance between delay and throughput.

3.2. Description of Core Components

3.2.1. Early Control Based on Queue Length (RED)

RED provides a probabilistic early drop mechanism based on the average queue length, enabling early detection of congestion trends, especially under bursty traffic. The drop probability is defined as in (1):

$$p_{RED} = \begin{cases} 0, & avg_{qlen} < min_{th} \\ \frac{avg_{qlen} - min_{th}}{max_{th} - min_{th}} \cdot p_{max}, & min_{th} \le avg_{qlen} \le max_{th} \\ 1, & avg_{qlen} > max_{th} \end{cases} \quad (1)$$

RED responds sensitively to increases in queue length and performs well with bursty traffic, but it becomes unstable if parameters are not properly configured.

3.2.2. Controlled Delay (CoDel)

The sojourn time of packet *i* is defined as: $\delta_i = t_{dequeue,i} - t_{enqueue,i}$. CoDel tracks the minimum sojourn time within each interval: $\delta_{min}(k) = \min_{i \in interval} \delta_i$

If $\delta_{min} > \delta_{target}$ over consecutive intervals, CoDel starts dropping packets with a decreasing interval based on the drop count (2):

$$t_{next_drop} = t_{last_drop} + \frac{interval}{\sqrt{count}} \quad (2)$$

https://doi.org/10.3390/engproc2026164011

CoDel does not require complex tuning and effectively mitigates bufferbloat; however, it reacts slowly to sudden traffic changes.

3.2.3. Proportional Integral Controller Enhanced (PIE)

PIE uses a PI controller to estimate and adjust the drop probability based on queuing delay. The queuing delay is estimated as $d_q(t) = \frac{\text{qlen (t)}}{\text{departure_rate (t)}}$. The drop probability is updated every interval T as follows (3):

$$p_{PIE}(t) = p_{PIE}(t - T) + \alpha\left(d_q(t) - d_{ref}\right) + \beta\left(d_q(t) - d_q(t - T)\right) \quad (3)$$

3.3. Proposed Hybrid AQM Based on Cross-Layer Multi-Agent Reinforcement Learning

The proposed MARL framework operates in a cross-layer manner, where the behaviors of RED, CoDel, and PIE are incorporated into the state and reward design. Instead of relying on these algorithms as independent control modules, the framework employs MetaPolicyNet as a unified learning-based controller that directly learns congestion management policies from network observations.

3.3.1. State Space Design

At each time step t, the global system state is represented by the vector (4):

$$\begin{aligned} s_t = \{&\text{qlen (t)}, \text{avg_qlen (t)}, \delta_{cur}(t), \delta_{min}(t), d_q(t), \text{ departure_rate (t)}, \\ &p_{RED}(t), p_{CoDel}(t), p_{PIE}(t), RTT(t), N_{flows}(t), \text{ ratio}_{\text{ elephant}}(t), ECN_{hist}(t)\} \end{aligned} \quad (4)$$

- Queue-related variables qlen (avg_{qlen}) reflect buffer occupancy. Both instantaneous and average values help detect short-term bursts and prevent overreaction, serving as key inputs for RED.
- Delay-related variables, including δ_{cur} (instantaneous sojourn time) and δ_{min} (minimum sojourn time per interval), distinguish persistent bufferbloat from transient congestion and form the basis of CoDel.
- The estimated delay and service rate, defined as $d_q(t) = \frac{\text{qlen}(t)}{\text{departure_rate (t)}}$, enable short-term congestion prediction and act as primary signals for the PIE controller.
- Cross-layer features such as RTT, number of flows, and the ratio of elephant flows provide end-to-end congestion awareness. Queue length alone is insufficient, while RTT helps differentiate congestion from long propagation paths.
- ECN history enables the agent to learn the responsiveness of TCP senders, avoiding unnecessary packet drops when ECN is sufficiently effective.

3.3.2. Action Space Design

At each step, the Network Agent outputs an action vector:

$$a_t = \{a_1(t), a_2(t), a_3(t), a_4(t)\} \quad (5)$$

with constraints $a_1 + a_2 + a_3 = 1$ and $a_i \in [0, 1]$. Here, a_1, a_2, and a_3 are weights for RED, CoDel, and PIE, while a_4 is an offset. This design ensures the agent transitions smoothly between strategies, avoids chattering during action selection, and keeps the resulting hybrid drop/mark probability p_{hybrid} bounded within the valid range $[0, 1]$, guaranteeing stable and well-behaved control decisions.

3.3.3. Hybrid Drop Probability

$$p_{\text{hybrid}}(t) = f_\theta(s_t) \quad (6)$$

This corresponds to $p_{hybrid}(t) = f_\theta(s_t)$, where MetaPolicyNet directly learns the optimal packet marking and dropping policy from the current network state. The learned policy replaces the independent control logic of RED, CoDel, and PIE, enabling adaptive congestion management under dynamic network conditions (6).

3.3.4. Reward Function

The reward function is defined as

$$r_t = -w_1 \cdot d_q(t) - w_2 \cdot \text{loss}(t) - w_3 \cdot \text{jitter}(t) + w_4 \cdot \text{throughput}(t) \quad (7)$$

where d_q directly reflects delay (QoE), packet loss penalizes excessive packet drops Hybrid AQM (MARL) Framework Architecture, jitter ensures stability for real-time and AI applications, and throughput maintains efficient resource utilization. The first three components $(-)$ queuing delay (d_q), packet loss rate $(\text{loss}(t))$, and jitter $(\text{jitter}(t))$—act as penalties, entering with negative signs, so the agent is penalized whenever delay increases, packets are dropped, or delay variation grows, discouraging behaviors that degrade service quality. Conversely, throughput is added positively as a reward, encouraging the agent to sustain high throughput. Because the weights satisfy $w_1 \gg w_2 \geq w_3, w_4 > 0$, the delay penalty dominates the reward signal, so the agent is driven primarily to mini-mize delay while the positive throughput term still discourages the collapse in through-put that a delay-only objective could otherwise cause; this design is intended to avoid an inverse trade-off between throughput and delay (7).

3.3.5. Proposed Hybrid AQM Framework Algorithm

- Description of the Proposed Hybrid AQM (MARL) Algorithm

To control congestion in a distributed manner, the router decides whether to drop or mark packets, while TCP (the sender) adjusts the congestion window (cwnd). These two sides influence each other, so a Hybrid-MARL approach is applied. During training, the framework adopts Centralized Training—Decentralized Execution (CTDE). The system uses the global state, s_t (entire network) for the Critic to learn the value function (8)

$$V(s_t) = \mathbb{E}\Big[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t\Big] \quad (8)$$

During deployment, each agent uses only local information to ensure practical implementation. Specifically, the Network Agent uses metrics such as queue length and delay, while the Transport Agent relies on RTT and ECN signals.

The policy is updated using Proximal Policy Optimization (PPO) to ensure stable learning, prevent overly large updates, and support continuous action spaces (9)

$$L^{PPO}(\theta) = \mathbb{E}\big[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\big] \quad (9)$$

- Hybrid AQM (MARL) Framework Architecture

Figure 1 illustrates the architecture of the Hybrid AQM-MARL framework, built on the Centralized Training with Decentralized Execution (CTDE) paradigm. The process begins at the Network Environment, where the network state is updated (State Update) and sent to two destinations: the Centralized Training block containing the Critic network, and directly to the Transport Agent. During training, the Critic receives Local Observations from the environment, computes them, and returns Advantage Feedback to both agents to refine their learned policies. Within the Decentralized Execution block, two agents operate independently: the Network Agent (functioning as Router/AQM) and the Transport Agent (functioning as TCP Sender), each performing its own Observation and producing a Local Action. The Network Agent's actions directly influence the Environment Impact through

Actions, while the actions of both agents are combined into a Reward, which feeds back to the Network Environment as Environment Reward—closing the training-execution loop and enabling dynamic adaptation to changing network conditions.

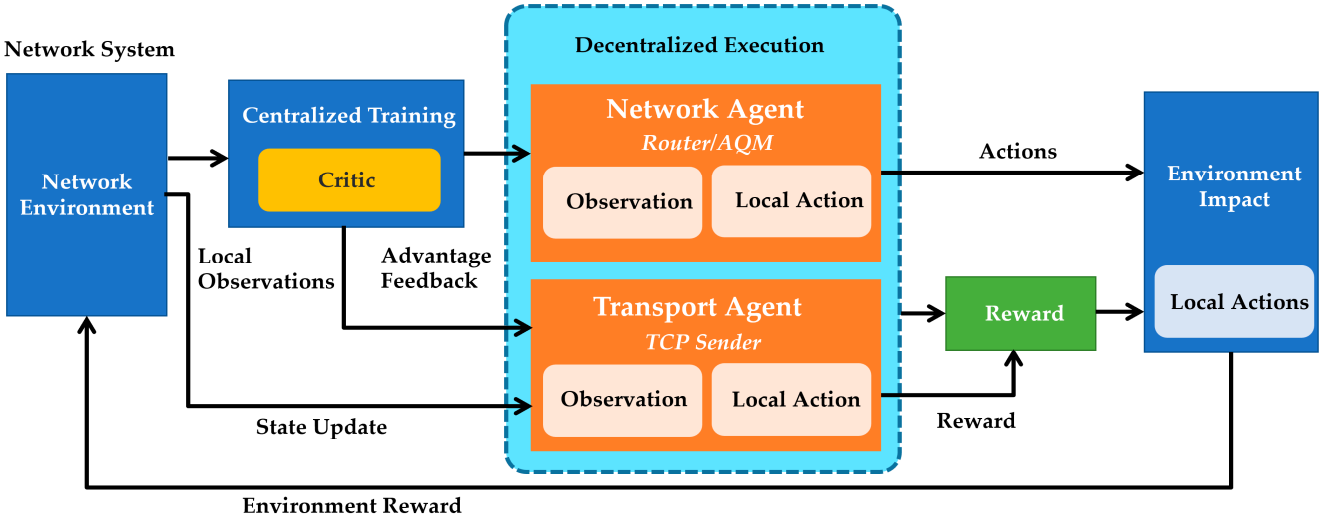


Figure 1. Architecture of the Proposed Hybrid AQM Framework.

4. Experiment

4.1. Experiment 1

The simulation runs on the NS-3 network simulator (ns-3 Consortium, Seattle, WA, USA) using a dumbbell topology with two routers (R1–R2). The bottleneck link has a bandwidth of 10 Mbps and a delay of 20 ms, while the access links connect to the routers with 100 Mbps bandwidth and 1 ms delay. The system generates mixed traffic, including TCP Bulk (4 flows), TCP Short (8 flows), UDP CBR (3 flows at ~2 Mbps), and UDP VoIP (4 flows at ~64 Kbps). The simulation duration is 120 s.

The generated data is processed to extract key features, including *Timestamp*, *packet_size*, *d_queue*, *queue_size*, *average_delay*, *queue_utilization*, *average_rtt*, and *ack_ratio*. The experimental dataset contains 815,895 samples and is used to evaluate the performance of AQM algorithms under high load conditions. The study evaluates multiple AQM algorithms, including RED, CoDel, PIE, ARED, FQ-CoDel, BLUE, ReCoPi (combining RED, CoDel, and PIE), and Hybrid-MARL.

4.2. Experiment 2

The UKMDDoSDN dataset comprises 1,820,910 samples of multi-flow data, including SYN flood, UDP flood, and ICMP flood attack methods, along with other background traffic, providing a diverse and realistic mixture of network traffic. The dataset was built on a realistic enterprise network topology, with an architecture comprising four isolated subnets: the External Network (h1: 192.168.10.x), the Enterprise Internal Network (h2–h5: 192.168.20.x), the Server/DMZ Network (h6: 192.168.30.x), and the Management Network (C0: 192.168.0.x), ensuring proper segmentation and accurately reflecting an enterprise environment. Owing to its large scale and diversity of attack types, this dataset serves as a suitable testbed for evaluating the adaptability and robustness of AQM algorithms under realistic DDoS attack conditions [19].

5. Results and Discussion

5.1. Evaluation of AQM Algorithm Parameters

5.1.1. Experimental Results Using Data Generated from NS-3 Simulations

After experimentation, the obtained results are as follows:

Table 2 summarizes the simulation results of eight AQM algorithms, showing that Hybrid-MARL comprehensively outperforms all other methods. In terms of delay, Hybrid-MARL achieves only 0.123 ms—the lowest in the table, a reduction of about 39% compared to RED (0.203 ms) and even lower than BLUE (0.153 ms), the second-best performer. For throughput, Hybrid-MARL reaches 465.595 Mbps, roughly 2.7 times higher than the average of the other schemes, and well above ReCoPi (399.702 Mbps), the second-highest. Its packet drop rate is only 1.202%, the lowest in the table, compared to 2.266–4.455% for the other algorithms. The Jain fairness index reaches 0.846—the highest value, reflecting more balanced resource allocation among flows. Link utilization is also the highest at 77.033%, while other algorithms range only from 39.478% to 53.463%. Notably, the Mean Reward of 10.8682, unique to Hybrid-MARL, reflects the agent's continuous self-learning and adaptation during training. Overall, these figures demonstrate that Hybrid-MARL simultaneously optimizes delay, throughput, fairness, and network resource efficiency.

Table 2. Summary table of AQM algorithm parameters from simulation data.

Parameters\Algorithm	Delay (ms)	Drop Rate (%)	ECN Rate (%)	Throughput (Mbps)	Jitter (ms)	Jain Fairness	Link Util (%)	Avg Queue (pkts)	Mean Reward
RED	0.203	3.797	3.274	158.785	0.260	0.627	43.603	59.26	N/A
CoDel	0.175	4.268	3.336	169.231	0.231	0.629	53.463	58.87	N/A
PIE	0.186	3.610	3.618	158.967	0.213	0.555	40.625	68.12	N/A
ARED	0.306	2.266	2.123	165.364	0.298	0.707	50.537	68.38	N/A
FQ-Codel	0.173	4.455	3.001	175.421	0.212	0.605	46.504	72.70	N/A
BLUE	0.153	3.588	1.663	155.545	0.238	0.591	44.967	55.60	N/A
ReCoPi	0.197	3.706	4.049	399.702	0.253	0.743	39.478	71.02	N/A
Hybrid-MARL	0.123	1.202	4.274	465.595	0.256	0.846	77.033	115.75	10.8682

5.1.2. Experimental Results with the UKMDDoSDN Dataset

Table 3 presents the experimental results on the UKMDDoSDN dataset, showing that Hybrid-MARL clearly outperforms the other seven AQM algorithms. For delay, Hybrid-MARL achieves only 1.064 ms, far lower than the 1.870–2.055 ms range of the other schemes, representing roughly a 43% reduction compared to RED. For throughput, Hybrid-MARL reaches 8852.380 Mbps—the highest in the table, well above PIE (6085.135 Mbps) and ReCoPi (6167.265 Mbps), the next-best performers. Its jitter is only 25.580 ms, notably lower than the 39.3–44.3 ms range of the other algorithms. Average queue length is also the shortest, at just 795.08 packets, compared to 1325–1450 packets for the others. Notably, link utilization reaches 98.050%, substantially higher than the 89.8–97.7% range of the other methods, indicating maximum bandwidth exploitation under DDoS attack conditions. The drop rate (0.456%) and ECN rate (0.000%) both remain low and stable. Most strikingly, the Mean Reward of 5519.3849 demonstrates the agent's superior learning and adaptation capability in a complex, adversarial network environment.

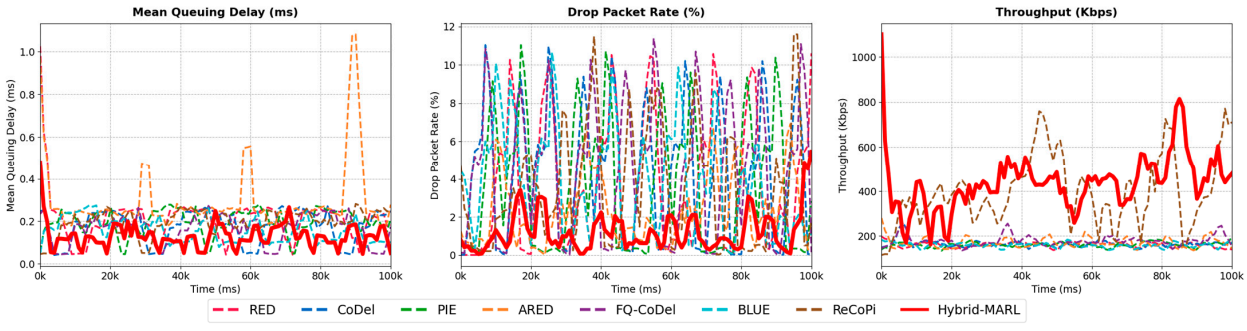
Table 3. Summary table of AQM Algorithm parameters from the UKMDDoSDN dataset.

Parameters\Algorithm	Delay (ms)	Drop Rate (%)	ECN Rate (%)	Throughput (Mbps)	Jitter (ms)	Jain Fairness	Link Util (%)	Avg Queue (pkts)	Mean Reward
RED	1.870	0.944	0.000	5723.725	41.371	1.061	92.110	1397.27	N/A
CoDel	1.961	0.253	36.273	5478.563	41.670	1.060	91.429	1339.76	N/A
PIE	1.896	1.012	0.003	6085.135	44.335	1.032	93.684	1355.79	N/A
ARED	2.055	0.983	0.001	5610.001	40.178	1.064	97.744	1325.68	N/A
FQ-CoDel	2.040	0.203	37.995	5757.070	42.993	1.012	89.867	1370.70	N/A
BLUE	1.897	0.534	22.060	6026.499	39.312	0.992	92.831	1449.92	N/A
ReCoPi	1.989	1.068	0.000	6167.265	41.131	1.046	92.622	1376.95	N/A
Hybrid-MARL	1.064	0.456	0.000	8852.380	25.580	1.037	98.050	795.08	5519.3849

5.2. Evaluating Network Performance Through Delay, Drop, and Throughput

5.2.1. Results Generated from the NS-3 Simulation

Figure 2 compares network performance between Hybrid-MARL and conventional congestion control algorithms in the NS-3 simulation environment. In subplot (a), Hybrid-MARL's mean queuing delay (red line) consistently stays below 0.2 ms and remains highly stable, whereas ARED exhibits sharp delay spikes exceeding 1.0 ms around the 30k, 60k, and 90k ms marks. Subplot (b) shows Hybrid-MARL's drop packet rate fluctuating at a low level, under 4%, in contrast to the other algorithms, which display large, cyclical swings peaking at 10–12%. Subplot (c) confirms Hybrid-MARL achieves the highest throughput, ranging from 400–600 Kbps with a gradual upward trend over time, roughly 2–3 times higher than RED, BLUE, and CoDel, which remain low and nearly flat (~150–200 Kbps). Overall, the simulation results confirm that Hybrid-MARL optimizes network resources and maintains superior system stability compared to the other schemes.



(**a**) (**b**) (**c**)

Figure 2. Comparing network performance between congestion control algorithms in the NS-3 simulation: (**a**) mean queuing delay; (**b**) packet drop rate; (**c**) throughput.

5.2.2. Results Generated from the UKMDDoSDN Dataset

Figure 3 clearly demonstrates that Hybrid-MARL substantially outperforms conventional AQM algorithms (RED, CoDel, PIE, ARED, FQ-CoDel, BLUE, ReCoPi) on the UKMDDoSDN dataset. In subplot (a), Hybrid-MARL's mean queuing delay stays around 1.0–1.2 ms, markedly lower than the 2.0–2.3 ms range of the other algorithms. Subplot (e) shows its jitter remaining the lowest and most stable throughout the simulation, without the repeated sawtooth spikes seen in the other methods. Subplot (c) confirms Hybrid-MARL achieves the highest throughput, driven by its fast adaptability to traffic fluctuations. Subplot (b) shows the drop packet rate tightly controlled around 0.45%, well below most competitors. Notably, subplot (d) shows an ECN mark rate near zero, and subplot (f)

shows a consistently small, stable average queue length. Overall, these results establish Hybrid-MARL as the most optimal and resilient approach to network resource management under DDoS conditions.

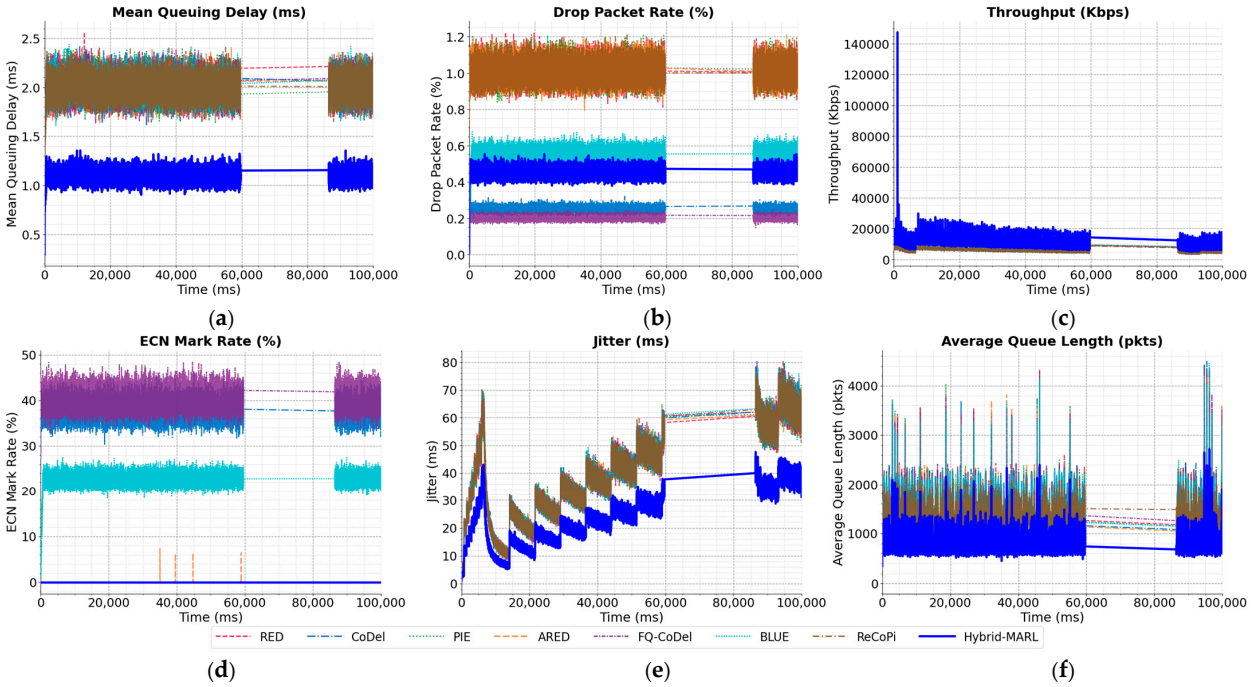


Figure 3. Comparing the network performance parameters of AQM algorithms on the UKMDDoSDN dataset: (**a**) mean queuing delay; (**b**) packet drop rate; (**c**) throughput; (**d**) ECN mark rate; (**e**) jitter; (**f**) average queue length.

Figure 4 illustrates the weight distributions selected by the PPO agent for the three components RED, CoDel, and PIE within the Hybrid-MARL framework. Subplot (b) shows CoDel receiving the dominant weight, with a mean of $\mu = 0.520$ and standard deviation $\sigma = 0.030$, confirming its core role in controlling queuing delay. In contrast, RED (a) and PIE (c) act only as supporting components, both with a much lower mean of $\mu = 0.240$ and $\sigma = 0.020$. The small standard deviations ($\sigma \approx 0.020$–0.030) across all three distributions indicate that the PPO agent converged to a stable policy, without large fluctuations across training episodes. These results demonstrate that MetaPolicyNet has learned a consistent strategy, prioritizing CoDel as the primary component while flexibly combining RED and PIE to optimize the overall performance of the Hybrid-MARL mode.

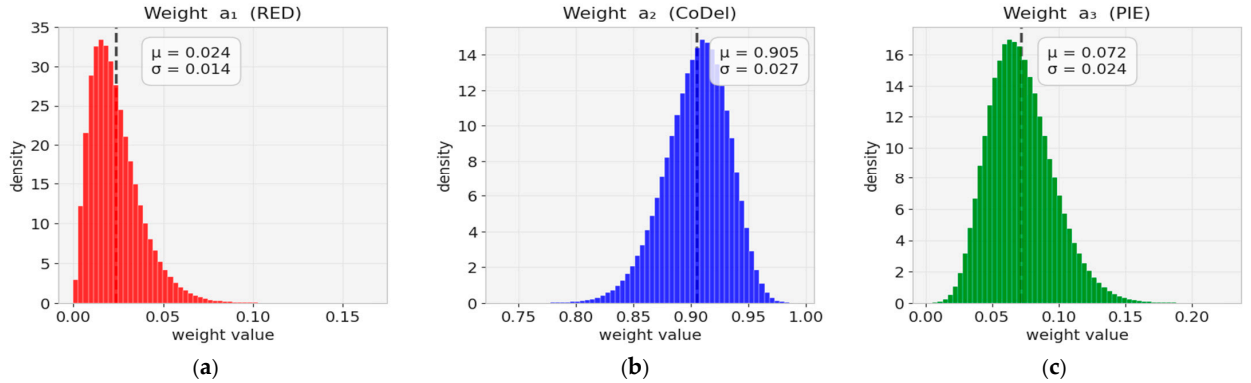


Figure 4. PPO-selected AQM Weight Distributions: (**a**) RED weight; (**b**) CoDel weight; (**c**) PIE weight.

https://doi.org/10.3390/engproc2026164011

In Hybrid-MARL, the reward function is designed to prioritize throughput, while the parameters of RED, PIE, CoDel, and ECN are optimized to reduce direct intervention in traffic management. At the same time, MetaPolicyNet is enhanced through architectural improvements, including dropout, additional network layers, Smooth Factor tuning, and adjustment of the α and β parameters. These enhancements enable MetaPolicyNet to serve as the central controller, completely replacing the independent agents of RED, PIE, and CoDel.

6. Conclusions and Recommendation

This study proposes Hybrid AQM-MARL, a cross-layer Active Queue Management (AQM) framework based on Multi-Agent Reinforcement Learning, employing Centralized Training with Decentralized Execution together with the PPO algorithm. Experimental results demonstrate that Hybrid AQM-MARL significantly outperforms both classical and modern AQM schemes (RED, CoDel, PIE, ARED, FQ-CoDel, BLUE, ReCoPi), achieving the lowest delay, the highest throughput, and substantially reduced packet drop rates. These findings confirm that integrating MARL with a cross-layer design spanning routers and end-hosts is an effective approach, enabling dynamic adaptation and comprehensive QoS optimization in complex network environments such as AI clusters, edge computing, and IoT systems.

Future work will focus on deploying the framework on a real-world testbed under larger-scale traffic conditions, integrating more advanced reinforcement learning methods (such as Offline RL and Multi-Objective RL), and optimizing computational cost to support implementation on hardware routers. Overall, Hybrid-MARL demonstrates strong potential as an intelligent queue management solution for next-generation networks.

Author Contributions: Conceptualization, V.X.C. and H.K.; methodology, V.X.C.; software, V.X.C.; validation, V.X.C., N.K.Q. and H.K.; formal analysis, V.X.C.; investigation, V.X.C.; resources, N.K.Q.; data curation, V.X.C.; writing—original draft preparation, V.X.C.; writing—review and editing, N.K.Q. and H.K.; visualization, V.X.C.; supervision, H.K.; project administration, N.K.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The UKMDDoSDN dataset analyzed in this study is publicly available in Mendeley Data at https://doi.org/10.17632/43nfv26gpk (accessed on 28 October 2026). The NS-3 simulation data generated in this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Toopchinezhad, M.P.; Ahmadi, M. Machine Learning Approaches for Active Queue Management: A Survey, Taxonomy, and Future Directions. *Comput. Netw.* **2025**, *262*, 111174. [CrossRef]
2. Floyd, S.; Jacobson, V. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.* **1993**, *1*, 397–413. [CrossRef]
3. Nichols, K.; Jacobson, V. Controlling Queue Delay (CoDel). *Commun. ACM* **2012**, *55*, 42–50. [CrossRef]
4. Pan, R.; Natarajan, P.; Piglione, C.; Prabhu, M.S.; Subramanian, V.; Baker, F.; VerSteeg, B. PIE: A lightweight control scheme to address the bufferbloat problem. In *Proceedings of the 2013 IEEE 14th International Conference on High Performance Switching and Routing (HPSR), Taipei, Taiwan, 8–11 July 2013*; IEEE: New York, NY, USA, 2013; pp. 148–155. [CrossRef]
5. Ali, M.H.; Öztürk, S. Efficient congestion control in communications using novel weighted ensemble deep reinforcement learning. *Comput. Electr. Eng.* **2023**, *110*, 108811. [CrossRef]

6. Bai, J.; Zhang, T.; Xie, G. MACC: Cross-layer multi-agent congestion control with deep reinforcement learning. *arXiv* **2022**, arXiv:2206.01972. [CrossRef]
7. Hoeiland-Jørgensen, T.; McKenney, P.; Taht, D.; Gettys, J.; Dumazet, E. "The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm." RFC 8290, IETF. January 2018. Available online: https://www.rfc-editor.org/info/rfc8290 (accessed on 30 October 2026).
8. De Almeida, L.C.; da Silva, W.R.D.; Tavares, T.C.; Pasquini, R.; Papagianni, C.; Verdi, F.L. Desired—Dynamic, enhanced, and smart ired: A p4-aqm with deep reinforcement learning and in-band network telemetry. *Comput. Netw.* **2024**, *244*, 110326. [CrossRef]
9. Naik; Mahendra, H. Quantifying Multimedia Streaming Quality: A Practical Analysis using PIE and Flow Queue PIE. *arXiv* **2025**, arXiv:2511.17525. [CrossRef]
10. Pan, R.; Prabhakar, B.; Psounis, K. CHOKe: A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation. In Proceedings of the IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064), Tel Aviv, Israel, 26–30 March 2000. [CrossRef]
11. Vijayaraj, A.; Bhuyan, H.K.; Raj, P.T.V.; Anand, M.V. Congestion avoidance using enhanced blue algorithm. *Wirel. Pers. Commun.* **2023**, *128*, 1963–1984. [CrossRef]
12. Kar, S.; Alt, B.; Koeppl, H.; Rizk, A. PAQMAN: A Principled Approach to Active Queue Management. *arXiv* **2023**, arXiv:2202.10352. [CrossRef]
13. Lhamo, O.; Ma, M.; Doan, T.V.; Scheinert, T.; Nguyen, G.T.; Reisslein, M.; Fitzek, F.H.P. RED-SP-CoDel: Random early detection with static priority scheduling and controlled delay AQM in programmable data planes. *Comput. Commun.* **2024**, *214*, 149–166. [CrossRef]
14. Mostafavi, S.; Roy, N.; Dán, G.; Gross, J. Active queue management with data-driven delay violation probability predictors. In Proceedings of the IEEE Global Communications Conference, Kuala Lumpur, Malaysia, 4–8 December 2023; pp. 6371–6376. [CrossRef]
15. Kim, M.; Jaseemuddin, M.; Anpalagan, A. Deep reinforcement learning based active queue management for iot networks. *J. Netw. Syst. Manag.* **2021**, *29*, 34. [CrossRef]
16. Pan, C.; Cui, X.; Zhao, C.; Wang, Y.; Wang, Y. An adaptive network congestion control strategy based on the change trend of average queue length (QCT-ARED). *Comput. Netw.* **2024**, *250*, 110566. [CrossRef]
17. Bachl, M.; Fabini, J.; Zseby, T. LFQ: Online learning of per-flow queuing policies using deep reinforcement learning. In *Proceedings of the 2020 IEEE 45th Conference on Local Computer Networks (LCN), Sydney, NSW, Australia, 16–19 November 2020*; IEEE: New York, NY, USA, 2020. [CrossRef]
18. Alhachem, C.; Kellil, M.; Bouabdallah, A. Towards a Multi-agent Deep Reinforcement Learning Approach for End-to-End Latency Minimization in Complex Communication Networks. In *International Conference on Advances in Computing Research*; Springer Nature: Cham, Switzerland, 2025. [CrossRef]
19. Bin Mokti, M.A. UKMDDoSDN: A DDoS Attacks Dataset for Software-Defined Networks. In *Mendeley Data*; Mendeley: London, UK, 2025; Volume V2. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.