Lightweight Automatic ECN Tuning Based on Deep Reinforcement Learning With Ultra-Low Overhead in Datacenter Networks

Jinbin Hu, *Member, IEEE*, Zikai Zhou, and Jin Zhang, *Member, IEEE*

***Abstract*—In modern datacenter networks (DCNs), mainstream congestion control (CC) mechanisms essentially rely on Explicit Congestion Notification (ECN) to reflect congestion. The traditional static ECN threshold performs poorly under dynamic scenarios, and setting a proper ECN threshold under various traffic patterns is challenging and time-consuming. The recently proposed reinforcement learning (RL) based ECN Tuning algorithm (ACC) consumes a large number of computational resources, making it difficult to deploy on switches. In this paper, we present a lightweight and hierarchical automated ECN tuning algorithm called LAECN, which can fully exploit the performance benefits of deep reinforcement learning with ultra-low overhead. The simulation results show that LAECN improves performance significantly by reducing latency and increasing throughput in stable network conditions, and also shows consistent high performance in small flows network environments. For example, LAECN effectively improves throughput by up to 47%, 34%, 32% and 24% over DCQCN, TIMELY, HPCC and ACC, respectively.**

***Index Terms*—Datacenter network, ECN, congestion control, deep reinforcement learning.**

I. Introduction

IN MODERN DCNs, with the increasingly stringent requirements for diverse services, such as big data processing [4], distributed storage [5], high-performance computing [6], and online services, effective congestion control is crucial to achieving ultra-low latency and high throughput. Explicit Congestion Notification (ECN) [7] becomes an essential congestion signal for mainstream congestion control mechanisms, which is widely enabled by commercial switches to indicate network congestion due to its simple and effective superior performance. By leveraging ECN, congestion control mechanisms quickly detect queueing building up and perform the corresponding rate adjustment to ensure efficient data transmission in DCNs.

However, a static ECN threshold is not sufficient to cope with dynamic changes in the network environment, leading to suboptimal performance for existing congestion control schemes. Furthermore, the preset static threshold is difficult to adapt to varying traffic patterns, resulting in a degradation of application performance. Additionally, network operators need to dedicate significant time and effort to setting suitable ECN thresholds in large distributed networks. Recently, dynamic threshold-setting has gained attraction in both academia and industry for datacenters. Dynamic ECN threshold-setting plays a crucial role in achieving optimal network performance in modern networks, especially in complex topologies with multiple paths. However, finding the right balance is challenging as a low threshold increases packet latency, while a high threshold leads to underutilization. Traditional threshold-setting algorithms do not work well in high-speed networks, further exacerbating underutilization. Moreover, bursty traffic in datacenters complicates dynamic ECN threshold-setting, necessitating adaptive algorithms to maintain optimal network performance. Therefore, advanced algorithms capable of adapting to changing traffic patterns are essential for setting dynamic ECN thresholds in high-speed networks with complex topologies.

To solve the drawbacks of the traditional static ECN threshold-setting, researchers have developed a new solution called ACC [8]. This technology leverages a deep learning-based approach to deploy DRL agents on each switch, allowing for autonomous and dynamic adjustments of ECN tagging thresholds based on real-time information of the buffer and traffic status. ACC significantly outperforms static ECN threshold-settings with zero configuration, achieving lower FCT and higher IOPS for storage services. In essence, ACC represents a breakthrough in addressing the challenges posed by traditional ECN threshold-settings. Although the RL-based model is promising, it requires significant computational resources. This poses a particular challenge for the RL-based automatic ECN threshold adjustments scheme, which operates on switches with limited and valuable CPU resources. When multiple concurrent traffic forwarding cases occur on the same switch, ACC may lead to severe performance degradation due to the large amount of reasoning overhead that interferes with the data path throughput and consumes non-negligible

Received 21 December 2023; revised 3 July 2024; accepted 23 August 2024. Date of publication 27 August 2024; date of current version 20 December 2024. This work is supported by the National Natural Science Foundation of China No. 62472050, the open research fund of Key Lab of Broadband Wireless Communication and Sensor Network Technology (Nanjing University of Posts and Telecommunications), Ministry of Education No. JZNY202308. A preliminary version of this paper appears in ICA3PP 2023 [1], Tianjin, China, October, 2023 [DOI: 10.1007/978-981-97-0798-0_19]. The associate editor coordinating the review of this article and approving it for publication was G. Leduc. *(Corresponding author: Jin Zhang.)*

Jinbin Hu is with the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, China, and also with the Key Laboratory of Broadband Wireless Communication and Sensor Network Technology, Ministry of Education, Nanjing University of Posts and Telecommunications, Nanjing 210049, China.

Zikai Zhou and Jin Zhang are with the School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410114, China (e-mail: jzhang@csust.edu.cn).

Digital Object Identifier 10.1109/TNSM.2024.3450596

CPU resources. One potential solution to reduce the overhead is to increase the time interval for automatically optimizing ECN configuration decisions, but this can result in serious performance degradation [9], [10].

We currently face a dilemma regarding ECN threshold auto-tuning using RL-based techniques. While these techniques are effective in adapting to different network circumstances, they suffer from high overhead issues that hinder their ability to quickly respond to changes in the dynamic network environment. Considering the aforementioned shortcomings of existing solutions, the question arises: Is there a solution that can dynamically provide the appropriate ECN threshold without incurring significant overhead costs?

In this paper, we propose a lightweight automatic ECN tuning mechanism called LAECN that provides a positive answer to the above question. LAECN utilizes a flexible hierarchical control architecture comprising decision and policy generator modules. The key design decision involves separating the resource-intensive policy module from the overall agent. LAECN establishes a hierarchical policy structure powered by DRL, which includes a decision module and a policy module. The policy module is activated only when the decision module determines that the current ECN threshold-setting is unsuitable for the environment. It then generates a new ECN threshold that is appropriate for the current situation. With LAECN, the overhead problem no longer poses a significant obstacle to the development of RL-based ECN auto-tuning algorithms.

The main contributions of this paper are as follows:

- This research focuses on congestion control in DCNs, with a specific emphasis on addressing the computational burden associated with automatically adjusting dynamic ECN thresholds on switches. The objective is to minimize computational overhead, thereby enhancing network performance and optimizing congestion control operations in DCNs.
- LAECN is a hierarchical and adaptive approach to control congestion that effectively addresses the limitations of static thresholds and computational overhead. By leveraging DRL, LAECN optimizes network functionality while minimizing the computational burden associated with dynamically adjusting ECN thresholds. LAECN is able to identify network environments dominated by small flows and dynamically adjust monitoring time intervals.
- The results of the simulation highlight the good performance of LAECN in bursty network environments, outperforming the ACC solution in both throughput and average flow complete time(AFCT). Under small flows scenarios, the enhanced LAECN shows more stable than HAECN [1] in terms of throughput. Compared to DCQCN, TIMELY, HPCC and ACC, LAECN significantly reduces the AFCT by up to 33%, 28%, 25%, and 21%, respectively. In addition, LAECN effectively achieves lower overhead than ACC.

II. Background and Motivation

Static ECN settings are incapable of adapting the network conditions change, which leads to suboptimal utilization of network resources and degraded performance. Tuning static ECN parameters is a complex and time-consuming task, and even if they are set optimally, a static threshold may not be suitable for all traffic types and congestion scenarios. Moreover, the use of static ECN can result in congestion collapse, especially in large datacenters, causing significant performance degradation and even network failure. Hence, an adaptive approach to ECN tuning is necessary to accommodate the dynamic nature of network conditions and traffic patterns, optimize network resource utilization, and ensure optimal performance.

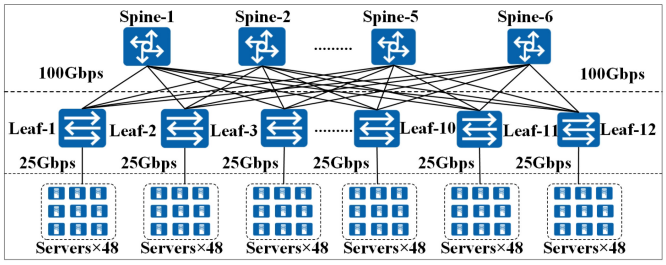


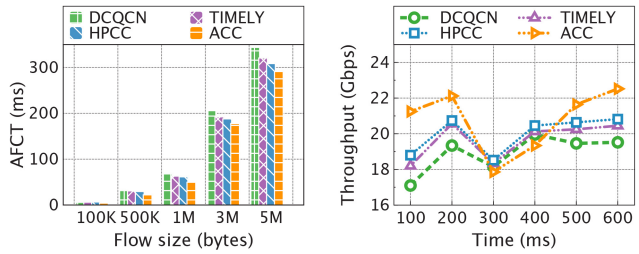
Fig. 1. Leaf-spine topology.

To meet the requirements of low latency and high bandwidth, HPCC [11] employs precise load information acquired from In-network telemetry (INT) to calculate accurate flow rates. Similarly, TIMELY [12] adjusts flow rates based on precise delay measurements using NIC timestamps instead of relying solely on ECN-based signals. These innovative designs have demonstrated significant performance enhancements. However, deploying these solutions in heterogeneous datacenters with legacy devices presents a challenge, as these devices may not support new features like INT.

An adaptive approach to ECN tuning is crucial for effective network congestion management. However, its implementation is constrained by the substantial computational resources it requires. This challenge is further exacerbated in the case of automatic RL-based ECN threshold tuning schemes, as they rely on limited CPU resources and can result in severe performance degradation. Based on these factors, it is evident that several challenging issues need to be addressed to achieve effective ECN tuning in modern datacenters.

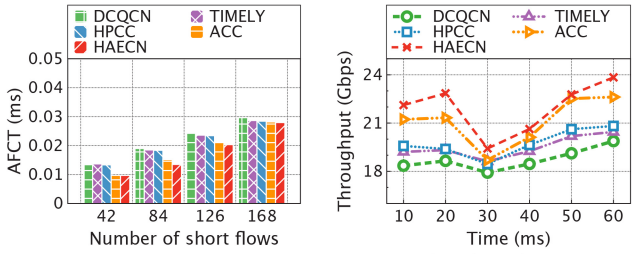
Observation 1: Sophisticated model inferences are often responsible for generating substantial computational overhead, which ultimately results in decreased system performance.

To evaluate the performance impact of the computational burden associated with a typical commercial switch with ACC functionality, we perform an extensive evaluation in a simulated network environment. This evaluation included a specific topology configuration featuring a leaf-spine architecture consisting of 12 leaf switches and 6 spine switches, as shown in Fig. 1. In this particular setup, each leaf switch is equipped with a total of 48 links, each with a bandwidth of 25 Gbps, to facilitate connectivity to the server infrastructure. In addition, both leaf and spine switches are interconnected by 6 links, each with a bandwidth of 100Gbps. Every fourth node linked to each leaf switch was programmed to generate a flow. These generated flows are subsequently directed towards the server that is specifically connected to leaf switch 1.



(a) In Bursty scenario (b) In gradual stable networks

Fig. 2. Performance under different networks environments.



(a) AFCT (b) Throughput

Fig. 3. Performance under small flows networks.

The analysis depicted in Fig. 2 (a) illustrates that the ACC algorithm's AFCT is adversely affected when the network experiences bursty links. It is important to highlight that while the ACC algorithm and the static threshold DCQCN [13] algorithm, as well as TIMELY and HPCC, exhibit certain advantages when dealing with small-scale traffic, their performance becomes comparable as the traffic size increases. In particular, the implementation of ACC with AFCT fails to adequately showcase the anticipated substantial advantages over the aforementioned RDMA control algorithms. These findings demonstrate that despite the potential benefits of employing DCQCN in conjunction with ACC, it introduces notable inference overhead, thereby consuming a significant amount of CPU resources. Consequently, this situation leads to considerable performance degradation, negatively impacting the overall efficiency of data transmission within the network infrastructure.

Observation 2: In stable DCNs environment, frequent adjustments to ECN threshold can disrupt network stability and lead to performance degradation.

In DCNs, it's common for the network environment to gradually stabilize over time. When an appropriate ECN threshold is set, there is often no need to frequently adjust it. However, the current approach for tuning ECN, known as ACC, doesn't take full advantage of this fact. Furthermore, ACC is not well-suited to handling the inherent uncertainty and variability in the network environment. The system may overreact to minor changes, resulting in unnecessary adjustments to the ECN threshold. This, in turn, can lead to further instability and congestion. Given these drawbacks, it's clear that a more efficient and effective approach for tuning ECN is needed in DCN environments. By taking advantage of the inherent stability of the network, it may be possible to reduce the overhead of ECN tuning and improve overall network performance.

The effectiveness of the ACC algorithm in a specific scenario, a series of experiments were conducted utilizing the NS-3 simulation framework [14]. The evaluation is carried out within the context of the Observation-1 scenario, with certain modifications introduced. Initially, the experimental setup involved the random selection of one flow per 12 nodes. However, after achieving network stabilization, the configuration is adjusted to direct one flow per 6 randomly selected nodes to a specific host connected to switch 1.

As illustrated in Fig. 2 (b), the ACC-loaded DCQCN algorithm exhibited a gradual stabilization process between $t_1 = 100ms$ and $t_2 = 200ms$, accompanied by a consistent increase in throughput. However, at time t_2, the ACC algorithm experiences pronounced network fluctuations as the traffic load intensifies. This leads to a continuous degradation in the overall network performance. Notably, within a certain range, the network throughput of the ACC algorithm is lower than that of other congestion control mechanisms such as DCQCN and TIMELY. These findings suggest that while the ACC algorithm effectively manages congestion during periods of network instability, it may generate redundant control decisions during stable periods, thereby resulting in performance degradation.

Observation 3: In a specific DCNs environment, the adjustment of ECN thresholds responds slowly and lags, leading to significant degradation of network performance. In the DCNs network, there are many bursty tiny flows smaller than 100KB. Typically, these small flows can be completed within a few RTTs, approximately 10 microseconds, much shorter than the current optimization method, which usually relies on multiple ACKs (e.g., 15 ACKs [3]). During this period, the system may exhibit latency and slow responses, causing the ECN thresholds to be inadequately adjusted in a timely manner, ultimately resulting in network instability and congestion. Given this limitation, there is a need for a more responsive ECN optimization method tailored to this specific scenario in the DCNs environment.

To evaluate the performance of the ACC and HAECN algorithms in this specific scenario, we conduct a series of NS-3 simulation experiments. The evaluation is performed in a topology consisting of 8 leaf switches and 4 spine switches. In this design, each leaf switch is equipped with a total of 24 links, each with a bandwidth of 25Gbps. The connections between the leaf switches and the spine switches are established through 8 links, each with a bandwidth of 100Gbps. In the experimental scenario shown in Fig. 3 (a), there are 42, 84, 126, and 168 small flows are generated in sequence between the source severs connected to leaf switches to the destination servers connected to Leaf Switch 1. In the experimental scenario depicted in Fig. 3 (b), 24 nodes within each server connected to the leaf switch randomly generated different small flows and transmitted them to servers connected to Leaf Switch 1.

The result in Fig. 3 (a) indicates that as the number of small flows in the network increases, both ACC and HAECN algorithms experience an adverse impact on the AFCT. The analysis of Fig. 3 (b) reveals that both ACC and HAECN

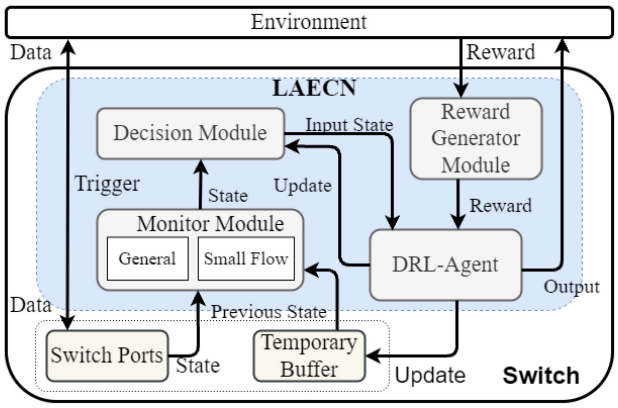


Fig. 4. LAECN overview.

algorithms undergo a decrease in throughput as the number of small flows increases, particularly at t1=30ms. These findings suggest that while ACC and HAECN may effectively manage congestion in typical stable networks, in an environment dominated by small flows, they may exhibit a slow response to changes, leading to a sharp decline in performance.

III. LAECN Design

A. LAECN Overview

LAECN is a hierarchical adaptive ECN threshold tuning approach for efficient network congestion control. By leveraging DRL,LAECN addresses the limitations of static threshold and computational overhead. In the architectural framework of LAECN, as is shown in Fig. 4, we need to entail the acquisition and observation of data as input to determine the RL agent's current state. Subsequently, a decision-making process is initiated to evaluate the necessity of activating the policy generator module, followed by appropriate actions being taken accordingly. Furthermore, the RL agent retrieves fresh state and reward values from the environment, facilitating the updating of its knowledge and optimizing its performance. This cyclical process ensures the continuous refinement and adaptation of the RL agent within the LAECN system.

B. Design Details of LAECN

The architecture of LAECN incorporates a hierarchical control logic, depicted in Fig. 5. Within each time interval, the monitoring module retrieves environmental data as input for the RL agent's current state. Subsequently, this data is transmitted to the decision module, where informed decisions are made, and the policy generator is periodically activated. It is important to highlight that the policy generator dynamically adjusts to triggers by updating the decision module accordingly. By employing this hierarchical control logic, LAECN effectively integrates data collection, decision-making, and policy adaptation, contributing to its overall efficacy in optimizing network performance while minimizing overhead.

Consequently, the policy generator remains idle when the ongoing ECN threshold performs satisfactorily, resulting in a considerably lower average activation frequency. Furthermore, owing to its straightforward learning objective, the monitor module is typically smaller in size compared to the policy generator. Consequently, the observer incurs a lesser regular computational cost than previous schemes based on DRL.

Monitoring Module: The monitoring module plays a crucial role in the RL agent by collecting real-time data from the network's switches. It continuously monitors the packet sizes for each link, initially considering all flows as short flows, which means it defaults to assuming the environment is in a short-flow state. When it detects that the data volume for a flow on the path exceeds a threshold of 100kb, it considers the environment to be in a long-flow state [2]. Additionally, it continuously monitors the queue length and output data rate for each link, we use normalized statistics to provide a more refined approach. It enables the agent to generalize observations from training sessions to unseen environments and improve the model's performance. Normalization also ensures equal treatment of input signals, avoiding the exaggerated impact of large values on the final model. Data collection occurs during consecutive monitoring intervals (Δt), with the monitoring module gathering real-time data, normalizing it to S_t, and transferring it to the decision module.

Decision Module: The decision module assesses the current ECN threshold in a dynamic network using a flag-triggering mechanism. If optimal performance can be achieved without the monitoring module's input, the decision module maximizes efficiency. When an update is needed, it triggers the policy generator for changes. State and action information $\{S^{save}, a^{save}\}$ are stored upon activation. After a time interval (Δt), with normalized data S_{t+1} from the monitoring module, the decision module repeats the process with $\{S_{t+1}, S^{save}\}$. When the monitored environment by the monitoring module is in a long-flow state, the default monitoring time Δt is set to 15 RTTs. When the monitored environment is in a short-flow state, the monitoring time is reduced to 5 RTTs to ensure more timely detection of minor changes in the network before small flows complete their transmission. This adjustment aims to address the system's lag caused by the rapid completion of transfers in a short-flow environment, which can lead to ineffective monitoring of network changes.

Policy Generator Module: The policy generator module is a key component of our RL agent. It uses a six-layer neural network (NN) model to determine optimal action values, denoted as $a_t = \{K_{\max}, K_{\min}, P_{\max}\}_t$, based on the decision module's input. These values adjust the ECN threshold-settings of switches. The RL agent employs a dynamic reward system, continuously updated with the latest data, to optimize performance. The system records the current state and actions, serving as reference points for the switches' ECN threshold-settings. This dynamic process enables the RL agent to navigate performance-affecting factors, adapt to changes, and minimize computational overhead and fluctuations. By avoiding unnecessary activation in stable network environments, the RL agent mitigates overreaction and prevents performance degradation.

C. Reinforcement Learning Agent in LAECN

RL is a powerful approach for dynamically tuning ECN and enabling adaptive system behavior. It utilizes the Markov

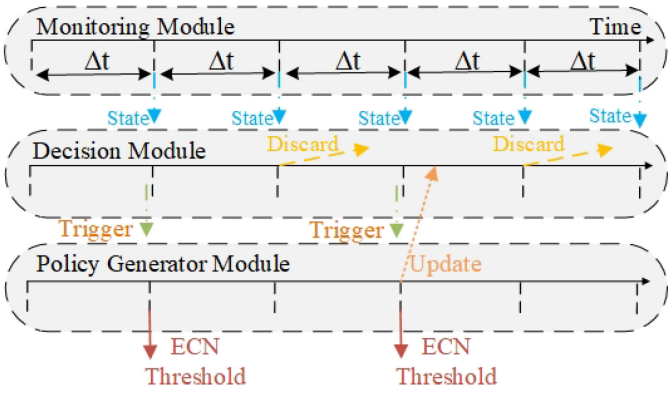


Fig. 5. The hierarchical control logic of LAECN.

Decision Process (MDP) with state space *S*, action set *A*, transition probability *P*, intermediate reward *R*, and discount factor γ. In the context of ECN tuning, RL divides time into monitoring intervals. At each slot, the RL agent observes the network state, takes action for ECN configuration, and receives a reward. The goal is to find the optimal policy that maximizes accumulated rewards. Our distributed RL agent design consists of independent agents at each switch, collecting local network information, making ECN configuration decisions, and updating policies. We employ Double-Deep Q-Learning (DDQN) and Deep Q-Learning (DQN) models to enhance ECN configurations, improve performance, and ensure scalability while minimizing computational overhead. By distributing RL agents across switches, we handle large-scale network complexities, adapt to changes, and optimize ECN configurations based on real-time information.

State: In RL, the term S_t refers to the input provided to the agent that describes the current environment. We use three important feature inputs, namely the normalized current port queue length (*Q*), the normalized link utilization (*T*), and the current ECN-setting ($Ecn = \{K_{\max}, K_{\min}, P_{\max}\}_t$) which serve as crucial indicators of the current network state. We further denote the combination of these features as the sub-state $S'_t = \{Q, T, Ecn\}_t$. To generalize the trend of changes across different network environments and evaluate queue length and throughput changes, we combine the sub-states from the past *k* timestamps and denote it as the current state $S_t = \{S'_{t-k+1}, S'_{t-k+2}, \ldots, S'_t\}$. Our agents utilize the information gathered from the environment to make informed decisions that enable consistently high network performance.

Normalization is crucial for ensuring comparable scale inputs in our approach. We calculate statistics by averaging values from the last report time to the current report time, reducing fluctuations. Link utilization ($T_{(t)}$) is obtained by dividing $txrate_{(t)}$ by the link capacity (*B*), representing the percentage of link capacity utilized. For queuing length normalization ($ql_{(t)}$), we use a step mapping function to map values to $Q_{(t)}$ between 0 and 1 [8]. Higher values indicate more severe congestion. This enables informed decision-making and improves network performance.

Action: In the paper, we define the action taken at time slot *t* as the configuration of the ECN setting, which includes the high marking threshold ($K_{\max}$), the low marking threshold ($K_{\min}$), and the tagging probability ($P_{\max}$).

$$a_t = \{K_{\max}, K_{\min}, P_{\max}\}_t \tag{1}$$

To reduce the complexity of the action space, we discretize the ECN adjustment action space and form a template for the ECN configuration at the switch. Specifically, we choose the discretization as the ECN marking threshold. By discretizing the action space, we can reduce the number of possible actions, making it easier for the agent to learn and make decisions. The choice of ECN marking threshold as the discretization allows us to maintain a fine granularity in the ECN adjustment action space, ensuring that we can make accurate and effective adjustments to the network.

Reward: DRL systems have shown great potential for improving the performance of various applications, including computer networks.In the context of network traffic control, the reward function plays a crucial role in defining the optimization objectives. It provides the necessary feedback to the agent on the effectiveness of its actions and helps improve the network's throughput and minimize queue length.

In this regard, we define the reward function for our network traffic control problem as a combination of the normalized link utilization and queue length, weighted by the parameter ω. Specifically, we calculate the reward as follows:

$$r = \left[\omega \times T_{(t)} + (1-\omega) \times Q_{(t)}\right] - \alpha \times \text{trigger} \tag{2}$$

where $T_{(t)}$ symbolizes the normalized representation of link utilization achieved by dividing the port rate by the link bandwidth. Moreover, $Q_{(t)}$ indicates a normalized queue mapping function that gradually decreases from 1 to 0, with lower values representing better queue conditions. The hyperparameter α (set to 0.05) determines the penalty's significance during model training [10]. A higher α reduces policy generator activation frequency, decreasing CPU overhead. The decision module learns to activate the policy generator judiciously based on α penalties. Reducing α allows more frequent trigger activation, achieving finer network control. Proper reward function design is crucial for desired performance in DRL-based network traffic control systems. This particular topic will be covered in more detail in the upcoming sections of the discussion.

D. Learning Algorithm in LAECN

ECN optimization can be represented as a DRL problem, which allows us to use advanced techniques to optimize the ECN protocol. Specifically, we use a DDQN to model our policy generator module, and a lightweight DQN network to model our decision module.

At time step *j*, our monitor module observes the environment and inputs S'_j to the decision module. The decision module integrates several consecutive time segments into S_j and decides whether the current ECN setting still satisfies the current environment, putting the trigger in the corresponding position. If the trigger is 0, the previously stored action a^{save} is executed. However, if the trigger is 1, S_j is sent to the policy generator module, which generates a new a_j. The tuple

Algorithm 1: Learning Algorithm in LAECN

Input: Replay Memory Buffer D, Batch Size N, Temporary Buffer M
Output: $a_t = \{K_{\max}, K_{\min}, P_{\max}\}_t$

for *every* Δt **do**
  Retrieve $S'_t = \{Q, T, Ecn\}_t$ **and obtain the current state**
  $S_t = \{S'_{t-k+1}, S'_{t-k+2}, \ldots, S'_t\}$;
  The trigger activation is based on $\{S, S^{\text{save}}\}$;
  if *triggering = 1* **then**
    The action a_t is selected as $\arg\max_a Q(S_t, a, \theta_i)$ and executed;
    Update M with the newest $\{S^{\text{save}}, a^{\text{save}}\}$;
  else
    continue;
  end
  At time step $t+1$, update S_{t+1}, r_t;
  Sample N transitions $\{S_j, a_j, r_j, S_{j+1}\}$ from D;
  $y_j = r_j + \gamma \times Q(S_{j+1}, \arg\max_a Q(S_{j+1}, a, \theta); \theta')$;
  $L(\theta) = \frac{1}{N}\sum_j \big(y_j - Q(S_j, a_j; \theta)\big)^2$;
  Compute the gradient for actors and critics: $\nabla_\theta L(\theta) = \frac{1}{N}\sum_j (y_j - Q(S_j, a_j; \theta))\nabla_\theta Q(S_j, a_j; \theta)$;
  Update the parameter θ;
end

(S_j, a_j, r_j, S_{j+1}), which includes the observation reward r_j and the next state S_{j+1}, is called experience. If the trigger is 1, we save the experience in buffer D for experience replay and update the saved state S_j^{save} and action a_j^{save} with S_j and a_j, respectively. The network is then trained by uniformly sampling from D. Periodic target updating and experience replay can significantly enhance and stabilize the training process of Q-learning.

This experience is stored in buffer D for experience replay, allowing the network to be trained by uniformly sampling from D. Periodic target updating and experience replay can significantly enhance and stabilize the training process of Q-learning, which is crucial for optimizing ECN. Overall, the use of DRL techniques allows us to formulate the ECN optimization problem in a new and powerful way and offers the potential for significant performance improvements in network congestion control.

IV. Implementation

The model architecture of our proposed method adopts a hierarchical policy model constructed using the Pytorch [15] framework. This hierarchical policy model is instrumental in facilitating effective decision-making and control in the network environment. To create a realistic and comprehensive training environment, we leverage the NS-3 network simulator, a widely used tool for network research and development. Additionally, we incorporate ns3-ai [19], a specialized interface, to establish a seamless connection between the NS-3 simulation environment and the agent model, enabling efficient training and evaluation.

TABLE I
Training Hyperparameters in LAECN

Hyperparameter	Value
Learning Rate	0.005
Gamma (γ)	0.98
Batch Size (N)	64
Model Update Interval	20
Monitoring Time Interval (Δt)	(5,15) ×RTT
Reward Penalty (α)	0.05

Hyperparameters play a crucial role in training machine learning models as they define the behavior and performance of the learning algorithm. In the context of LAECN, our proposed method, we carefully select and tune specific hyperparameters to ensure effective training and optimization. The hyperparameters listed in Table I provide important insights into the configuration of LAECN during the training process.

The reward punishment coefficient (α) determines the step size for adjustments in each iteration of the algorithm. We chose $\alpha = 0.05$ based on extensive experimental results, indicating that this value ensures system stability while allowing for relatively fast convergence across various network environments and traffic patterns.

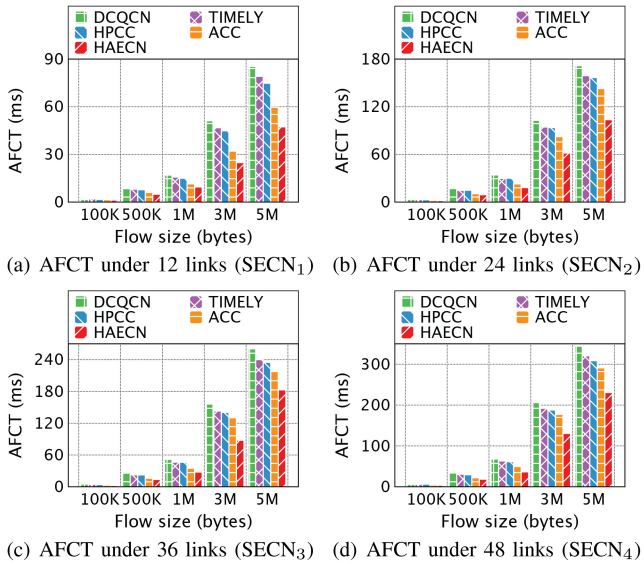
The discount factor (Gamma) balances the importance of immediate rewards versus future rewards. A Gamma value of 0.98 means we value long-term gains while not neglecting short-term benefits. Experimental validation showed that Gamma = 0.98 provided the optimal balance in our application scenarios.

The smaller step sizes can prevent excessive oscillations during the training process while providing enough flexibility to adapt to dynamic network changes. When choosing the Gamma value, we considered the system's latency characteristics and the temporal effectiveness of decisions. A Gamma value close to 1 indicates a higher emphasis on future rewards, which is crucial in dynamic environments.

V. Performance Evaluation

In this section, our objective is to assess the effectiveness and practical applicability of LAECN by utilizing NS-3 simulations. These simulations serve as a valuable tool for evaluating and comparing various factors pertaining to performance and computational overhead. Through the application of NS-3 simulations, we can accurately quantify the impact of LAECN on service delivery and thoroughly evaluate its relevance in real-world scenarios.

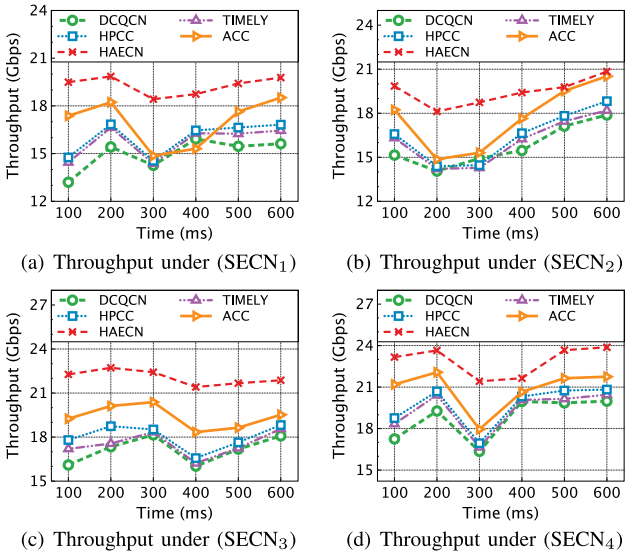
To assess the effectiveness of LAECN in typical DCNs, we conduct a large-scale simulation using the NS-3 framework. The objective is to create a realistic network environment, which is achieved by implementing a two-level leaf-spine topology comprising 12 leaf switches and 6 spine switches. Our evaluation focuses on four distinct network environments, denoted as SECN_1, SECN_2, SECN_3, and SECN_4, respectively. These environments share a similar topology but differ in the



(a) AFCT under 12 links ($SECN_1$) (b) AFCT under 24 links ($SECN_2$)

(c) AFCT under 36 links ($SECN_3$) (d) AFCT under 48 links ($SECN_4$)

Fig. 6. Performance under different bursty network schemes.



(a) Throughput under ($SECN_1$) (b) Throughput under ($SECN_2$)

(c) Throughput under ($SECN_3$) (d) Throughput under ($SECN_4$)

Fig. 7. Performance under gradual stabilization network scenario.

number of links connected to each leaf switch. Specifically, we vary the number of links per leaf switch, examining scenarios with 12, 24, 36, and 48 links. Each link in the network has a bandwidth capacity of 25 Gbps. Additionally, to ensure efficient communication, we interconnect the leaf and spine switches using 6 links with 100Gbps.

A. Performance Under Bursty Scenario

To enhance the randomness of our experimental network and ensure generality across the four aforementioned network scenarios, namely $SECN_1$, $SECN_2$, $SECN_3$, and $SECN_4$, we implement specific configurations.These configurations involve programming every fourth node connected to each leaf switch to generate a random data flow. The purpose of this setup is to introduce variability in the network traffic patterns and simulate a realistic data transmission scenarios. Furthermore, the generated flows are directed towards the server connected to leaf switch 1, allowing us to analyze the impact of such data flows on the performance of the network and the effectiveness of the evaluated mechanisms.

As shown in Fig. 10, even with the continuous increase of leaf nodes, the performance of ACC and LAECN far exceeds other advanced load congestion control mechanisms in $SECN_1$ and $SECN_2$. Notably, the AFCT achieved by ACC and LAECN is found to be shorter than DCQCN, TIMELY and HPCC. Meanwhile, due to overcoming the high overhead of ACC communication, LAECN has achieved low latency and much higher performance than ACC. Specifically, LAECN reduces the AFCT by about 30%, 24%, 22% and 16% compared to DCQCN, TIMELY, HPCC and ACC at flow size of 5M in the case of 36 leaf nodes.

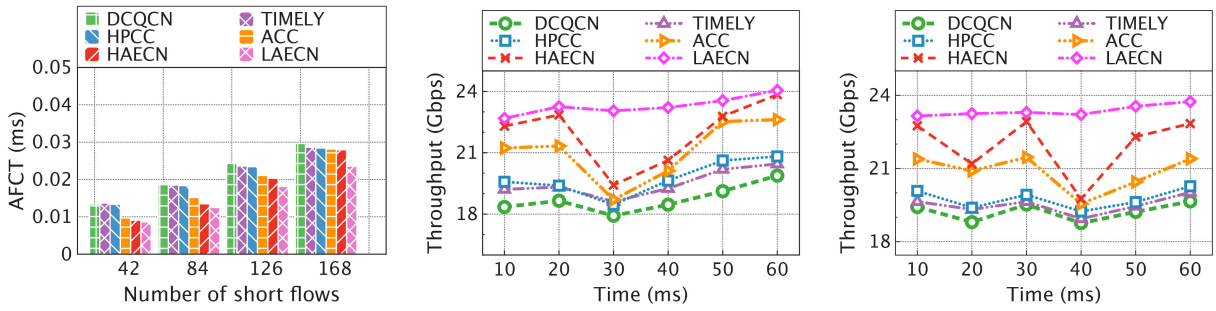
Meanwhile, it is important to highlight that employing ACC with the DCQCN algorithm introduces a significant computational overhead when the number of links on the leaf switches increases to 48 in $SECN_4$, as illustrated in Fig. 10. This overhead becomes particularly evident when the network operates at high speeds. Consequently, the network experiences an increase in transmission delay and a noticeable decline in overall performance. Specifically, compared to DCQCN, TIMELY, HPCC and ACC at flow size of 5M, LAECN can reduce the AFCT by 33%, 28%, 25% and 21%, respectively. Interestingly, even in $SECN_3$, where ACC is implemented, a relatively high AFCT is observed, surpassing that of algorithms such as DCQCN and TIMELY.

In Fig. 6, it can be seen that LAECN can effectively reduce the average FCT. This is because LAECN adopts a hierarchical design structure, effectively isolating the resource intensive policy generator module from the main agent. This design ensures that the policy generator module remains inactive when the ongoing ECN thresholds are deemed satisfactory, resulting in a significant reduction in the average activation frequency. The policy generato module is activated only when the decision module determines that the current ECN threshold-setting is unsuitable for the network environment. This approach allows LAECN to consistently maintain high throughput and low queuing latency, enabling the network to operate efficiently.

B. Performance Under Stable Network Environments

In DCNs, it is common for the network environment to gradually stabilize over time, transitioning from an initial unstable state to a more predictable and steady state. In our paper, we focus on assessing link characteristics, specifically bandwidth and link latency, to understand how different congestion control mechanisms perform in stable network environments.

To assess the effectiveness of the LAECN algorithm in the small flows environment, we conduct a comprehensive series of experiments. These evaluations are carried out within the context of the aforementioned scenario, incorporating specific modifications. Initially, the experimental setup involves the randomized selection of one data flow per 12 nodes. However, upon achieving network stability, we adjust the configuration to direct one data flow per 6 randomly selected nodes towards a designated host connected to switch 1, while maintaining



(a) AFCT under small flows scenarios (b) Throughput under ($SECN_2$) (c) Throughput under ($SECN_4$)

Fig. 8. Performance under different small flows scenario.

consistency in the other aspects of the design. The objective of these adjustments is to observe and analyze the performance of the LAECN algorithm under specific conditions, particularly its impact on the network dynamics and congestion control capabilities.

Fig. 7 illustrates the emergence of the ACC, indicating its gradual stabilization and the subsequent increase in throughput. Over time, the ACC algorithm begins to surpass other congestion control mechanisms like DCQCN and TIMELY. Nevertheless, as the traffic load intensifies, the ACC algorithm experiences significant network fluctuations. Consequently, the overall network performance undergoes a continuous oscillatory decline. Notably, the ACC algorithm's throughput remains lower than that of DCQCN and TIMELY within a specific range. These observations imply that while the ACC algorithm may generate unnecessary control decisions during stable periods, leading to performance degradation. In contrast, LAECN exhibits remarkable performance in a stable environment. Its network throughput steadily grows and surpasses state-of-the-art RDMA algorithms such as DCQCN, without significant fluctuations or disruptions. Specifically, compared to DCQCN, TIMELY, HPCC and ACC under different leaf nodes, LAECN increases throughput up to 47%, 34%, 32%, 24%, respectively.

LAECN algorithm demonstrates remarkable efficacy in achieving high throughput while maintaining low queue lengths. Neglecting to promptly adjust the current ECN threshold in response to increasing queue lengths can result in rapid queue accumulation and subsequent latency spikes. In contrast, LAECN proactively responds to queue length and link utilization changes by employing lower ECN thresholds to generate more ECN-tagged packets. Conversely, as the queue length approaches a lower threshold, LAECN applies a higher ECN threshold to prevent potential starvation and ensure optimal throughput performance. This dynamic adjustment of the ECN marker threshold by LAECN ensures the maintenance of short queues and adaptability to prevailing environmental conditions.

The flexible hierarchical control architecture of LAECN proves advantageous as the network environment stabilizes. This architecture effectively separates the policy module from the overall agent, allowing for informed decision-making regarding the network environment before making direct modifications to the ECN thresholds. This approach ensures system stability and prevents misadjustment of the DCN network environment caused by sudden changes in ECN thresholds, which could potentially degrade the performance of the entire network.

Our evaluation demonstrates the excellent performance of LAECN in a stabilized network environment. It consistently achieves high throughput and significantly low latency, thereby ensuring stable and efficient network operations. These results indicate that LAECN presents a promising solution for optimizing network performance in stable datacenter networks, surpassing ACC in terms of stability, congestion control, and overall performance.

C. Performance Under Small Flows Scenarios

To assess the performance of the LAECN algorithm in an environment dominated by small flows, we conduct a series of comprehensive experiments. We use a topology comprising 8 leaf switches and 4 spine switches. In this design, each leaf switch is equipped with 24 links, each with a bandwidth of 25Gbps. The connections between the leaf switches and the spine switches are established through links with a bandwidth of 100Gbps. In the experiment Scenario 1, each server connected to a leaf switch randomly generate 42, 84, 126, and 168 small flows, directed towards servers connected to Leaf Switch 1. In the experiment Scenario 2, each server connected to the leaf switch randomly generate different small flows within its 24 nodes and transmitted them to servers connected to Leaf Switch 1. These adjustments were made to observe and analyze the performance of LAECN in a specific environment, particularly the evaluation in a network environment with a significant number of small flows.

In the experiment Scenario 1, as shown in Fig. 8 (a), in a network environment with a small number of small flows, ACC, HAECN, and LAECN consistently exhibit high performance. However, as the number of small flows increased, ACC and HAECN experience a decline in performance due to their lagging response in this environment. Particularly in the environment with 168 small flows, LAECN achieves an average AFCT reduction of approximately 23%, 21%, 21%, 19%, and 18% compared to DCQCN, TIMELY, HPCC, ACC, and HAECN, respectively.

In the experiment Scenarios 2, as depicted in Fig. 8 (b) and Fig. 8(c), with an increasing number of small flows, ACC and HAECN show a sharp decrease in throughput at t1=30ms

TABLE II
FLOW SIZE DISTRIBUTION OF REALISTIC WORKLOADS

	Web Server	Cache Follower	Web Search	Data Mining
0-10KB	63%	50%	49%	78%
10KB-100KB	18%	3%	3%	5%
100KB-1MB	19%	18%	18%	8%
$>$ 1MB	0	29%	20%	9%
Average flow size	64KB	701KB	1.6MB	7.41MB

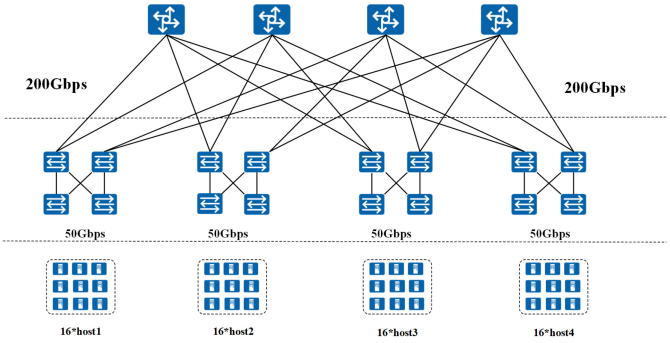


Fig. 9. Fat-tree Topology.

in $SECN_2$, and in $SECN_4$, ACC and HAECN experienced throughput decreases at t2=20ms and t3=40ms. LAECN addresses the issue of delayed response by adjusting the monitoring time Δt, consistently exhibiting high performance.

As shown in Fig. 11 (b), ACC, HAECN, and LAECN all maintain high performance under a small amount of small traffic. However, when t1=40ms and t2=100ms suddenly increase the sending of a large number of small traffic, ACC and HAECN will lead to a sharp decline in throughput due to delayed response. This additional experiment can prove that LAECN still has good performance under pressure.

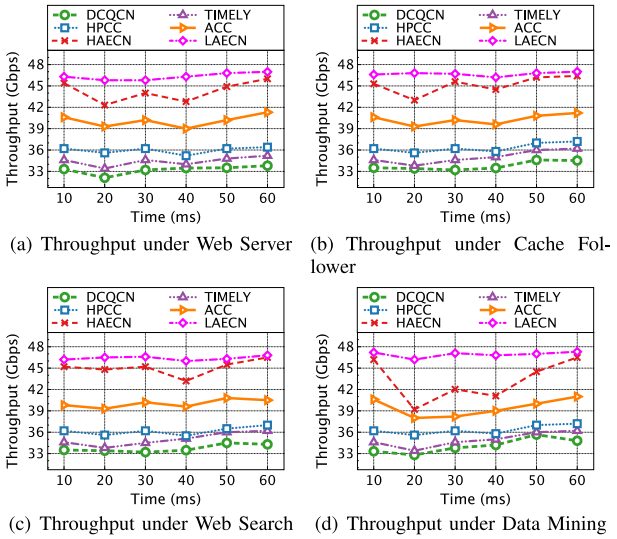
These results indicate that LAECN provides a viable solution in an environment dominated by small flows.

D. Performance Under Heterogeneous Workloads Scenarios

Due to today's increasingly complex network topologies, it is necessary to test in large data center networks with heterogeneous workloads. We choose to use the fat tree network topology with k=4. In this topology, there are 4 switches in the core layer, 16 switches in the core layer and the aggregation layer, and 64 servers at the bottom. Furthermore, the specific topology is presented in Fig. 9. Table II shows the distribution ratio of traffic under four distinct traffic modes.

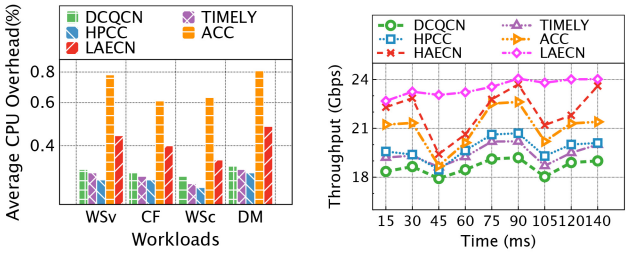
We use four workloads with the same distributions as the realistic ones, including Web Server (WSv), Cache Follower (CF), Web Search (WSc) and Mining (DM) [31], [32], [33], which cover a wide range of average flow sizes from 64KB to 7.41MB and more than half of flows are less than 10KB.

Fig. 10 present above indicates how we examin the throughput variations of the fat-tree topology under diverse traffic scenarios. Through observation, it is found that in the two experimental scenarios consisting of a large number of small flows, namely Web Server and Data Mining, HAECN and



(a) Throughput under Web Server (b) Throughput under Cache Follower
(c) Throughput under Web Search (d) Throughput under Data Mining

Fig. 10. Throughput under four different workloads.



(a) Average CPU overhead under different workloads (b) Throughput of pressure test

Fig. 11. Performance under different networks environments.

ACC exhibit significant throughput fluctuations, while LAECN maintains a consistently high throughput. Meanwhile, in the two scenarios of Cache Follower and Web Search, the throughput fluctuations of HAECN and ACC are not as pronounced, but LAECN still maintains the same high throughput.

Due to the layered control structure of LAECN, the decision module makes an informed choice before the ECN threshold is modified, whether in a stable network environment or in a small-stream dominated one. This not only guarantees the stability of the system but also avoids the disturbance to the DCN network environment caused by the sudden alteration of the ECN threshold.

As shown in Fig. 11 (a), we calculate the average CPU overhead of all mechanisms in four scenarios: Web Server, Cache Follower, Web Search, and DataMining. We can find that ACC has significantly higher overhead than traditional congestion control mechanism due to its deep reinforcement learning module, while LAECN has a flexible hierarchical control architecture to avoid the CPU overhead increase caused by ACC overreacting due to network changes. Compared to ACC, LAECN reduced CPU overhead by an average of about 31%.

VI. RELATED WORK

Congestion Control in DCNs: Congestion control has remained a prominent and enduring research focus within

the networking domain for over three decades. Contemporary congestion control mechanisms, such as DCTCP [20], [21], DCQCN, and their enhanced iterations, heavily rely on the utilization of the ECN mechanism to facilitate rate control. The ECN mechanism plays a pivotal role in sustaining high-performance DCNs. However, conventional approaches predominantly employ static methods for setting the ECN thresholds, which lack the necessary adaptability to effectively address dynamic fluctuations in network conditions. This limitation often leads to suboptimal performance outcomes.

Previous Work Related to ECN: Extensive investigations have been conducting to enhance latency and throughput performance in modern datacenter networks through the careful determination of ECN marking thresholds. According to ECN* [22], optimizing the instant queue length-based ECN threshold can lead to the attainment of optimal incast performance through the implementation of RED-like probabilistic marking. In a similar vein, TCN proposes the utilization of the sojourn time, which quantifies the duration that packets reside in the queue, as a means to label packets. Moreover, the study presented in ECN# focuses on analyzing the variation of RTT within the datacenter network and marks packets based on both instantaneous and persistent congestion states. It is important to highlight that despite the availability of two threshold parameters ($K_{\max}$ and $K_{\min}$) for the ECN switch, a significant number of researchers have commonly opted to assign identical values to both thresholds. Consequently, these studies primarily focus on the examination of a single threshold rather than exploring the potential benefits of leveraging distinct threshold values.

Learning-based Network Optimization: Learning-based methods have become popular for optimizing network performance and setting parameters for congestion control mechanisms. Remy [23], Indigo [24], Vivace [25], and Aurora [26] are examples of such methods, using techniques like dynamic rate adjustment and DRL. Orca combines traditional TCP Cubic with learning-based approaches to tackle unpredictable traffic patterns. While most learning-based approaches focus on adjusting sending rates based on feedback, ACC proposes using DRL to autonomously adjust ECN parameters. ACC has shown promising results in reducing FCT and maintaining high throughput, but still faces challenges related to overhead and stability in stable network environments.

VII. Conclusion

This paper presents LAECN, a novel methodology for addressing congestion control and determining optimal ECN thresholds in DCNs. Unlike existing approaches, LAECN leverages DRL techniques to dynamically adapt ECN thresholds based on the evaluation conducted by its decision modules. A key contribution of LAECN lies in its adoption of a flexible hierarchical control architecture, which encompasses decision and policy generation modules to effectively reduce computational overhead. This adaptive framework ensures superior network performance without incurring unnecessary computational burden, while also guaranteeing network stability in small flows network environments. Experimental evaluations demonstrate the superior performance of LAECN compared to existing methods, showcasing its ability to significantly reduce computational overhead and enhance network functionality. Specifically, LAECN demonstrates remarkable efficiency by reducing latency and significantly increasing throughput compared to DCQCN, TIMELY, HPCC, and ACC across various leaf nodes, with throughput improvements up to 47%, 34%, 32% and 24%, respectively.

Future enhancements for LAECN include incorporating advanced machine learning models, improving real-time adaptability, enhancing scalability, and integrating robust security features. Efforts are also underway to integrate LAECN with other data center management systems such as automatic traffic routing, energy optimization mechanisms, centralized network management platforms, and hybrid cloud environments. These enhancements and integrations aim to provide more comprehensive and efficient network management, ensuring optimal performance, reduced operational costs, and improved scalability for modern data centers.

References

[1] J. Hu et al., "HAECN: Hierarchical automatic ECN tuning with ultra-low overhead in datacenter networks," in *Proc. IEEE ICA3PP*, 2023, pp. 324–343.

[2] J. Hu, J. Huang, W. Lv, Y. Zhou, J. Wang, and T. He, "CAPS: Coding-based adaptive packet spraying to reduce flow completion time in data center," *IEEE/ACM Trans. Netw.*, vol. 27, no. 6, pp. 2338–2353, Dec. 2019.

[3] M. Mathis and J. Mahdavi, "Forward acknowledgement: Refining TCP congestion control," in *Proc. ACM SIGCOMM*, 1996, pp. 281–291.

[4] Y. Zhao, Y. Huang, K. Chen, M. Yu, S. Wang, and D. Li, "Joint VM placement and topology optimization for traffic scalability in dynamic datacenter networks," *Comput. Netw.*, vol. 80, pp. 109–123, Apr. 2015.

[5] C. Bunnag, P. Jareoncharsri, P. Tantilipikorn, P. Vichyanond, and R. Pawankar, "Epidemiology and current status of allergic rhinitis and asthma in Thailand—ARIA Asia-Pacific workshop report," *Asian–Pacific J. Allergy Immunol.*, vol. 27, no. 1, pp. 79–86, 2009.

[6] X. Lu et al., "High-performance design of Hadoop RPC with RDMA over InfiniBand," in *Proc. ICPP*, 2013, pp. 641–650.

[7] K. Ramakrishnan, S. Floyd, and D. Black, "The addition of explicit congestion notification (ECN) to IP," RFC 3168, IETF, 2001.

[8] S. Yan, X. Wang, X. Zheng, Y. Xia, D. Liu, and W. Deng, "ACC: Automatic ECN tuning for high-speed datacenter networks," in *Proc. ACM SIGCOMM*, 2021, pp. 1–14.

[9] S. Abbasloo, C.-Y. Yen, and H. J. Chao, "Classic meets modern: A pragmatic learning-based congestion control for the Internet," in *Proc. ACM SIGCOMM*, 2020, pp. 632–647.

[10] H. Tian, X. Liao, C. Zeng, J. Zhang, and K. Chen, "Spine: An efficient DRL-based congestion control with ultra-low overhead," in *Proc. CoNEXT*, 2022, pp. 261–275.

[11] Y. Li, M. Alizadeh, M. Yu, R. Miao, and F. Kelly, "HPCC: High precision congestion control," in *Proc. ACM Special Interest Group Data Commun.*, 2019, pp. 44–58.

[12] R. Mittal et al., "TIMELY: RTT-based congestion control for the datacenter," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 537–550, 2015.

[13] Y. Zhu et al., "Congestion control for large-scale RDMA deployments," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 523–536, 2015.

[14] "Network simulator," Apr. 2023. [Online]. Available: https://wwwnsnam.org

[15] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, p. 721.

[16] J. Hu et al., "Load balancing with multi-level signals for lossless datacenter networks," *IEEE/ACM Trans. Netw.*, vol. 32, no. 3, pp. 2736–2748, Jun. 2024.

[17] J. Hu, Y. He, W. Luo, J. Huang, and J. Wang, "Enhancing load balancing with in-network recirculation to prevent packet reordering in lossless data centers," *IEEE/ACM Trans. Netw.*, early access, May 27, 2024, doi: 10.1109/TNET.2024.3403671.
[18] G. Zeng, W. Bai, G. Chen, K. Chen, D. Han, and Y. Zhu, "Combining ECN and RTT for datacenter transport," in *Proc. APNet*, 2017, pp. 36–42.
[19] H. Yin et al., "NS3-AI: Fostering artificial intelligence algorithms for networking research," in *Proc. Workshop NS-3*, 2020, pp. 57–64.
[20] M. Alizadeh et al., "Data center TCP (DCTCP)," in *Proc. ACM SIGCOMM*, 2010, pp. 63–74.
[21] M. Alizadeh, A. Javanmard, and B. Prabhakar, "Analysis of DCTCP: Stability, convergence, and fairness," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 39, no. 1, pp. 73–84, 2011.
[22] H. Wu, J. Ju, G. Lu, C. Guo, Y. Xiong, and Y. Zhang, "Tuning ECN for data center networks," in *Proc. Emerg. Netw. Exp. Technol.*, 2012, pp. 25–36.
[23] K. Winstein and H. Balakrishnan, "TCP ex machina: Computer-generated congestion control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 123–134, 2013.
[24] F. Y. Yan et al., "Pantheon: The training ground for Internet congestion-control research," in *Proc. Annu. Tech. Conf.*, 2018, pp. 731–743.
[25] M. Dong et al., "PCC vivace: Online-learning congestion control," in *Proc. NSDI*, 2018, pp. 343–356.
[26] N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar, "A deep reinforcement learning perspective on Internet congestion control," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1–10.
[27] R. Xu, W. Li, K. Li, X. Zhou, and H. Qi, "DarkTE: Towards dark traffic engineering in data center networks with ensemble learning," in *Proc. ACM IWQOS*, 2021, pp. 1–10.
[28] Y. Liu, W. Li, W. Qu, and H. Qi, "BULB: Lightweight and automated load balancing for fast datacenter networks," in *Proc. ACM ICPP*, 2022, pp. 1–11.
[29] W. Li, X. Yuan, K. Li, H. Qi, and X. Zhou, "Leveraging endpoint flexibility when scheduling coflows across geo-distributed datacenters," in *Proc. IEEE INFOCOM*, 2018, pp. 873–881.
[30] W. Li, S. Chen, K. Li, H. Qi, R. Xu, and S. Zhang, "Efficient online scheduling for coflow-aware machine learning clusters," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2564–2579, Oct.–Dec. 2022.
[31] X. He, W. Li, S. Zhang, and K. Li, "Efficient control of unscheduled packets for credit-based proactive transport," in *Proc. ICPADS*, 2023, pp. 1–13.
[32] S. Abbasloo, C. Y. Yen, and H. J Chao, "Wanna make your TCP scheme great for cellular networks? Let machines do it for you!" *IEEE J. Sel. Areas Commun.*, vol. 39, no. 1, pp. 265–279, Jan. 2021.
[33] J. Hu, J. Huang, Z. Li, J. Wang, and T. He, "A receiver-driven transport protocol with high link utilization using anti-ECN marking in data center networks," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 2, pp. 1898–1812, Jun. 2023.
[34] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, "Homa: A receiver-driven low-latency transport protocol using network priorities," in *Proc. ACM SIGCOMM*, 2018, pp. 221–235.
[35] P. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, and S. Shenker, "pHost: Distributed near-optimal datacenter transport over commodity network fabric," in *Proc. ACM CoNEXT*, 2015, pp. 1–12.
[36] I. Cho, K. Jang, and D. Han, "Credit-scheduled delay-bounded congestion control for datacenters," in *Proc. ACM SIGCOMM*, 2017, pp. 1–14.
[37] A. Amanov, A. Majidi, N. Jahnabakhsh, and A. Çetin, "Adjusting ECN marking threshold in multi-queue DCNs with deep learning," *J. Supercomput.*, vol. 79, pp. 5443–5468, Mar. 2023.
[38] A. Majidi, X. Gao, S. Zhu, N. Jahanbakhsh, J. Zheng, and G. Chen, "MiFi: Bounded update to optimize network performance in software-defined data centers," *IEEE/ACM Trans. Netw.*, vol. 31, no. 1, pp. 322–335, Feb. 2023.
[39] A. Majidi, N. Jahanbakhsh, X. Gao, J. Zheng, and G. Chen, "ECN+: A marking-aware optimization for ECN threshold via per-Port in data center networks," *J. Netw. Comput. Appl.*, vol. 152, Feb. 2020, Art. no. 102504.
[40] A. Majidi, N. Jahanbakhsh, X. Gao, J. Zheng, and G. Chen, "DC-ECN: A machine-learning based dynamic threshold control scheme for ECN marking in DCN," *Comput. Commun.*, vol. 150, pp. 334–345, Jan. 2020.
[41] A. Majidi, X. Gao, N. Jahanbakhsh, J. Zheng, and G. Chen, "Priority policy in multi-queue data center networks via per-port ECN marking," in *Proc. IMCOM*, 2020, pp. 1–8.
[42] A. Majidi, X. Gao, N. Jahanbakhsh, S. Jamali, J. Zheng, and G. Chen, "Deep-RL: Deep reinforcement learning for marking-aware via per-port in data centers," in *Proc. ICPADS*, 2019, pp. 392–395.

Jinbin Hu (Member, IEEE) received the B.E. and M.E. degrees from Beijing Jiao Tong University, China, in 2008 and 2011, respectively, and the Ph.D. degree in computer science from Central South University, China, in 2020. She is currently a Postdoctoral researcher with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, and also with the School of Computer and Communication Engineering, Changsha University of Science and Technology, China. Her current research interests are in the area of datacenter networks, RDMA networking, and learning-based network systems.

Zikai Zhou is currently pursuing the M.E. degree with the School of Computer and Communication Engineering, Changsha University of Science and Technology, China. His research interests are in the area of datacenter networks.

Jin Zhang (Member, IEEE) received the master's degree from Hunan University in 2004, and the Ph.D. degree from Zhejiang University in 2007. He is currently a Professor with Changsha University of Science and Technology. With over 100 publications in international journals and conferences, his research focuses on software engineering and artificial intelligence. Additionally, he serves as the Secretary General of the Changsha Branch of China Computer Society CCF, a Deputy Secretary-General of Hunan Computer Society, and a Vice Chairman of the Computer Education Committee of Hunan Higher Education Association.