HAECN: Hierarchical Automatic ECN Tuning with Ultra-low Overhead in Datacenter Networks*

Jinbin Hu, Youyang Wang, Zikai Zhou, Shuying Rao, Rundong Xin, Jing Wang, and Shiming He

School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410004, China
{jinbinhu,znwj_cs,smhe_cs}@csust.edu.cn
{1437891086a}@gmail.com
{zhouzikai,shuyingrao,rundongxin}@stu.csust.edu.cn

Abstract. In modern datacenter networks (DCNs), mainstream congestion control (CC) mechanisms essentially rely on Explicit Congestion Notification (ECN) that is widely supported by commercial switches 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 Automatic ECN Tuning algorithm (ACC) dynamically adjusts the ECN threshold based on reinforcement learning (RL). However, the RL-based model consumes a large number of computational resources, making it difficult to deploy on switches. In this paper, we present a hierarchical automated ECN tuning algorithm called HAECN, which can fully exploit the performance benefits of deep reinforcement learning with ultra-low overhead. The simulation results show that HAECN improves performance significantly by reducing latency and increasing throughput in stable network conditions. For example, HAECN effectively improves throughput by up to 47%, 34%, 32% and 24% over DCQCN, TIMELY, HPCC and ACC, respectively.

Keywords: Datacenter Network · ECN · Congestion Control · Deep Reinforcement Learning.

1 INTRODUCTION

In modern DCNs, with the increasingly stringent requirements for diverse services, such as big data processing [1], distributed storage [2], high-performance

* Shiming He is the corresponding author.
This work is supported by the National Natural Science Foundation of China (62102046, 62072056), the Natural Science Foundation of Hunan Province (2023JJ50331, 2022JJ30618, 2020JJ2029), the Hunan Provincial Key Research and Development Program (2022GK2019), the Scientific Research Fund of Hunan Provincial Education Department (22B0300).

computing [3], and online services, effective congestion control is crucial to achieveing ultra-low latency and high throughput. Explicit Congestion Notification (ECN) [4] 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 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 [5]. This technology leverages a deep learning-based approach to deploy DRL agents on each switch, allowing for autonomous and dynamic adjustment 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 adjustment 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 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 [6,7].

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 hierarchical automated ECN tuning mechanism called HAECN that provides a positive answer to this question. HAECN 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. HAECN 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 HAECN, 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.
- HAECN is a hierarchical and adaptive approach to network congestion control that effectively addresses the limitations of static thresholds and computational overhead. By leveraging DRL, HAECN optimizes network functionality while minimizing the computational burden associated with dynamically adjusting ECN thresholds.
- The results of the simulation highlight the exceptional performance of HAECN in gradual stabilization network environments, surpassing the ACC solution in both throughput and round-trip time (RTT). For instance, HAECN 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.

The rest of the paper is organized as follows. We establish our design motivation in Section 2. In Section 3, we provide an overview of the design and introduce the details of HAECN. Section 4 discusses the implementation. The simulation results are presented in Section 5. We present the related works in Section 6 and conclude the paper in Section 7.

2 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.

To meet the requirements of low latency and high bandwidth, HPCC [8] employs precise load information acquired from In-network telemetry (INT) to calculate accurate flow rates. Similarly, TIMELY [9] 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.

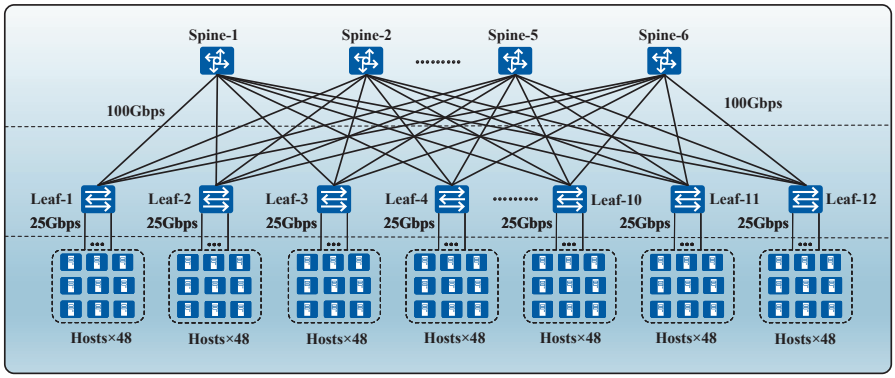


Fig. 1. Leaf-spine topology

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 data flow. These generated flows are subsequently directed towards the server that is specifically connected to leaf switch 1.

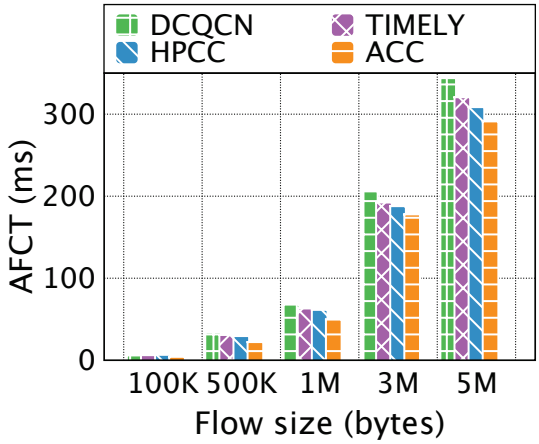


Fig. 2. Performances under bursty scenario

The analysis depicted in Fig. 2 illustrates that the ACC algorithm's average FCT is adversely affected when the network experiences bursty links. It is important to highlight that while the ACC algorithm and the static threshold DCQCN [10] 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 per-

formance degradation, negatively impacting the overall efficiency of data transmission within the network infrastructure.

Observation 2: In stable DCNs evironment, 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.

To assess the effectiveness of the ACC algorithm in a specific scenario, a series of experiments were conducted utilizing the NS-3 simulation framework [11]. The evaluation is carried out within the context of the Observasion-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.

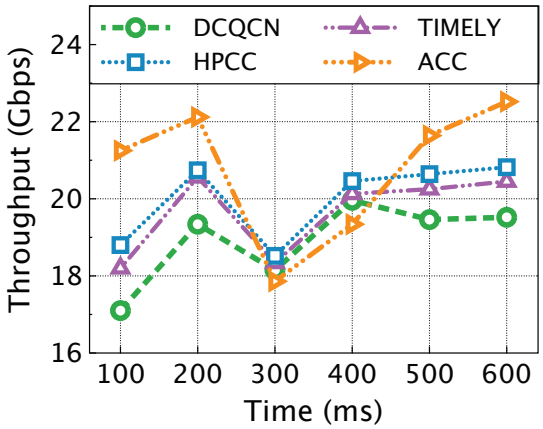


Fig. 3. Performance under gradual stable networks

As illustrated in Fig. 3, 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.

3 HAECN DESIGN

3.1 HAECN Overview

HAECN is a hierarchical adaptive ECN threshold tuning approach for efficient network congestion control. By leveraging DRL, HAECN addresses the limitations of static threshold and computational overhead. In the architectural framework of HAECN, 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 HAECN system.

3.2 Design Details of HAECN

The architecture of HAECN 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, HAECN 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 queue length and output data rate for each link. Instead of raw data, we use normalized statistics to provide a more refined approach. The normalization

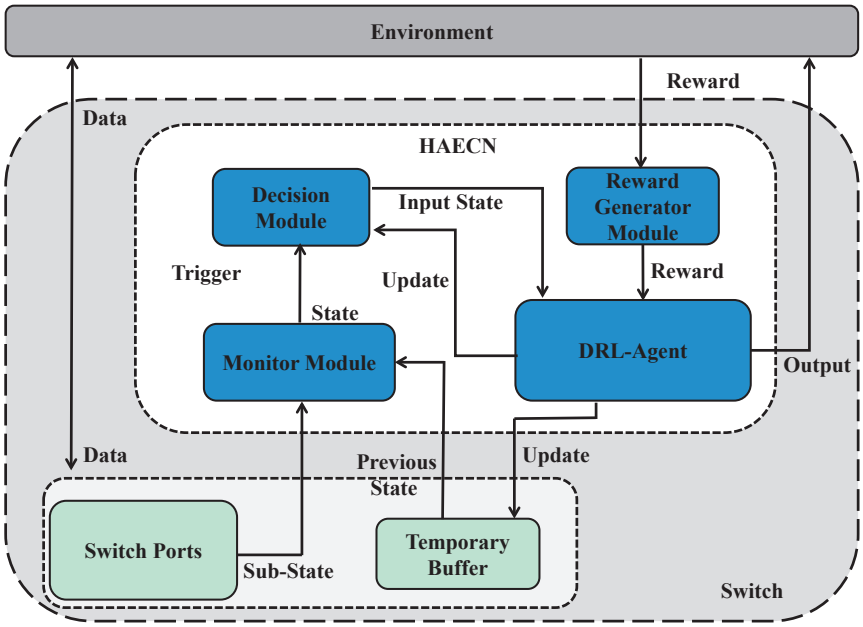


Fig. 4. HAECN overview

process is detailed in §3.3. 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 $\{S_{t+1}, S^{Save}\}$.

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

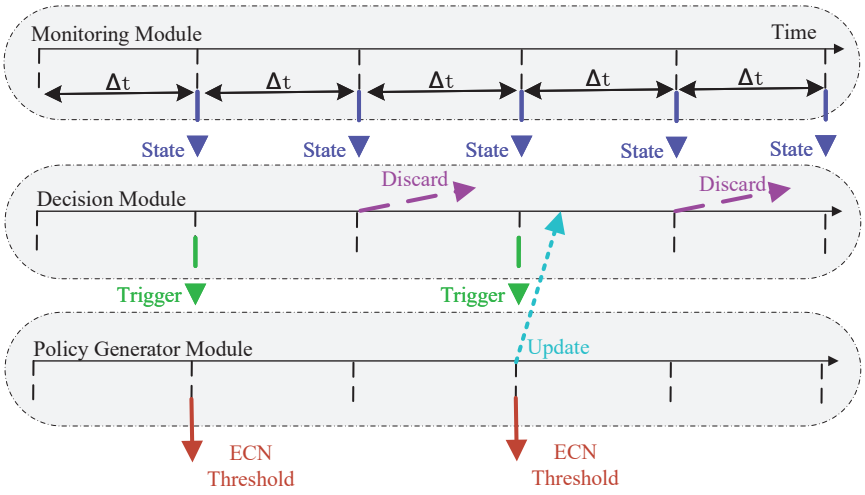


Fig. 5. The hierarchical control logic of HAECN

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.

3.3 Reinforcement Learning Agent in HAECN

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

tion 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 [5]. 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. The reward function is a critical factor that significantly impacts the performance of a DRL system. 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:

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

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 [7]. 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.

3.4 Learning Algorithm in HAECN

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.

Algorithm 1: HAECN's Learning Algorithm

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 sub-state $S'_t = \{Q, T, Ecn\}_t$ **and obtain the current state**
    The agent retrieves a sub-state $S'_t = \{Q, T, Ecn\}_t$ and obtains the current state;
  $S_t = \{S'_{t-k+1}, S'_{t-k+2}, \ldots, S'_t\}$;
  The trigger activation is determined 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 the Temporary Memory Buffer with the newest $\{S^{\text{Save}}, a^{\text{Save}}\}$;
  else
    continue;
  end
  At time step $t+1$, observe S_{t+1}, r_t, and store the transition;
  Sample N transitions $\{S_j, a_j, r_j, S_{j+1}\}$ from D;
  $y_j = r_j + \gamma \times Q(S_{j+1}, \arg\max_a Q(S_{j+1}, a, \theta); \theta')$;
  $L(\theta) = \frac{1}{N} \sum_j (y_j - Q(S_j, a_j; \theta))^2$;
  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 parameters θ;
end

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 (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.

4 IMPLEMENTATION

The model architecture of our proposed method adopts a hierarchical policy model constructed using the Pytorch [12] 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 [13], a specialized interface, to establish a seamless connection between the NS-3 simulation environment and the agent model, enabling efficient training and evaluation.

Table 1. Training hyperparameters in HAECN

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

For the training and evaluation of our proposed method, we utilize a Linux computer equipped with an NVIDIA GeForce RTX 3090 GPU. This high performance computing infrastructure enables us to efficiently train and evaluate our HAECN model. Leveraging the computational power of the GPU, we can effectively handle the complex computations and optimizations required during the

training process. By employing this state-of-the-art hardware setup, we ensure accurate and reliable results for our HAECN model.

Hyperparameters play a crucial role in training machine learning models as they define the behaviour and performance of the learning algorithm. In the context of HAECN, our proposed method, we carefully select and tune specific hyperparameters to ensure effective training and optimization. The hyperparameters listed in Table 1 provide important insights into the configuration of HAECN during the training process.

5 EVALUATION

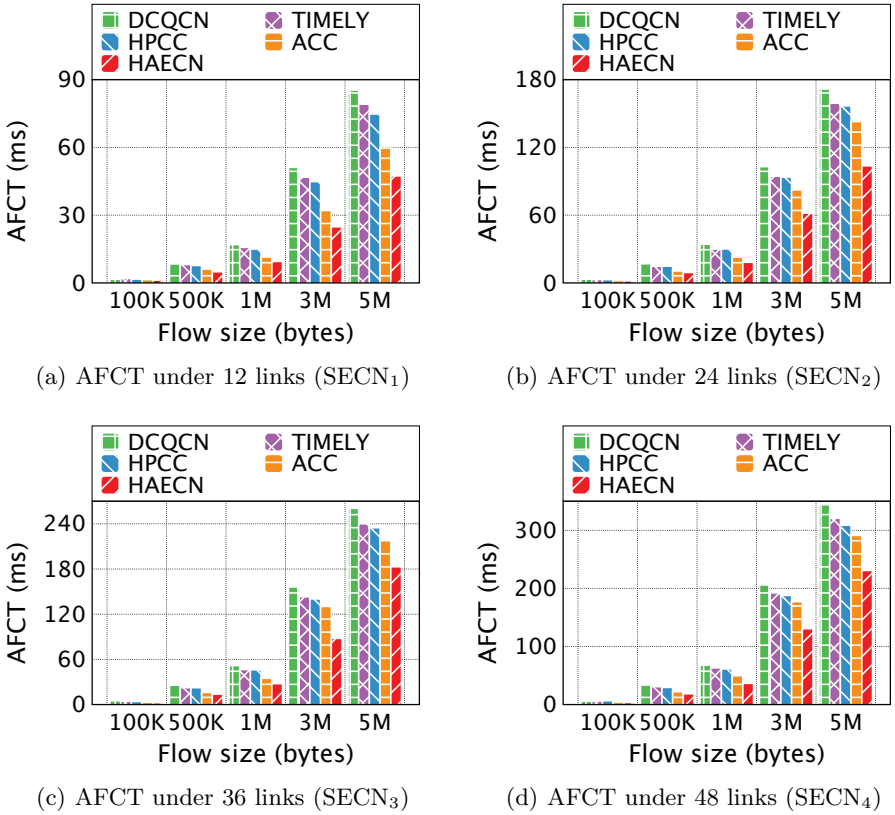
In this section, our objective is to assess the effectiveness and practical applicability of HAECN by utilizing NS-3 simulations. These simulations serve as a valuable tool for evaluating and comparing various factors pertaining to performance and computational overhead. Through the application of NS-3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

To assess the effectiveness of HAECN 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 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, each supporting a bandwidth of 100 Gbps.

5.1 Performance Under Bursty Networks

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. 6, even with the continuous increase of leaf nodes, the performance of ACC and HAECN far exceeds other advanced load balancing mechanisms in SECN_1 and SECN_2. Notably, the average FCT achieved by ACC and HAECN is found to be shorter than DCQCN, TIMELY and HPCC. Meanwhile, due to overcoming the high overhead of ACC communication, HAECN



(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

has achieved low latency and much higher performance than ACC. Specifically, HAECN reduces the average FCT 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. 6. 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, HAECN can reduce the average FCT by 33%, 28%, 25% and 21%, respectively. Interestingly, even in $SECN_3$, where ACC is implemented, a relatively high average FCT is observed, surpassing that of algorithms such as DCQCN and TIMELY.

In Fig. 6, it can be seen that HAECN can effectively reduce the average FCT. This is because HAECN adopts a hierarchical design structure, effectively iso-

lating 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 HAECN to consistently maintain high throughput and low queuing latency, enabling the network to operate efficiently.

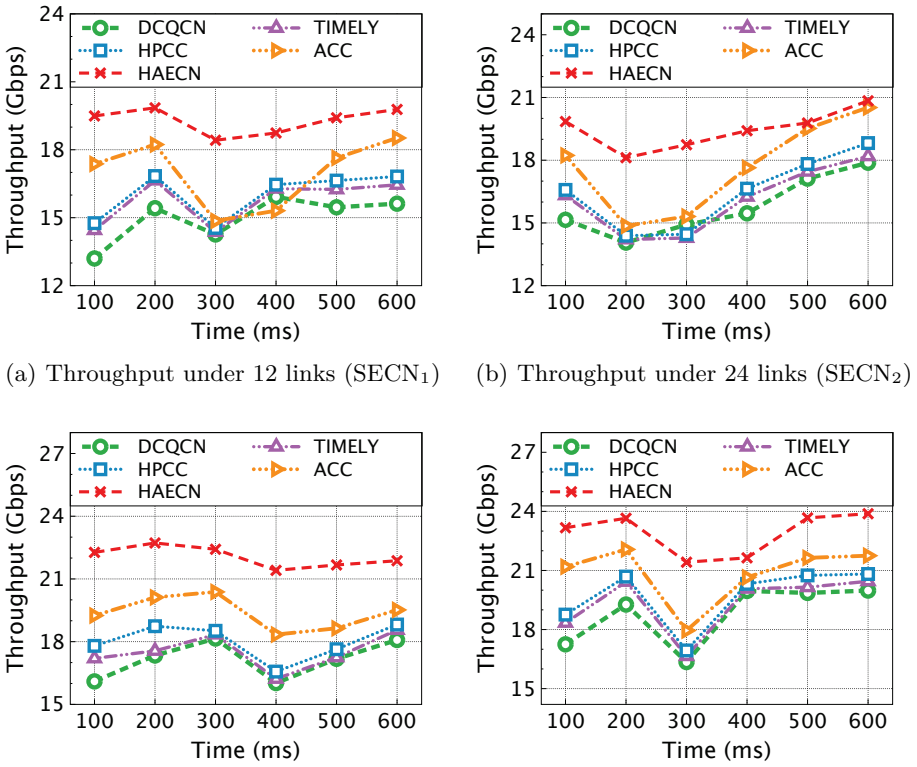
5.2 Performance Under Gradual Stabilization Network Environments

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

Fig7 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, HAECN 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, HAECN increases throughput up to 47%, 34%, 32%, 24%, respectively.

HAECN 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



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

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

Fig. 7. Performance under gradual stabilization network environments

queue accumulation and subsequent latency spikes. In contrast, HAECN 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, HAECN applies a higher ECN threshold to prevent potential starvation and ensure optimal throughput performance. This dynamic adjustment of the ECN marker threshold by HAECN ensures the maintenance of short queues and adaptability to prevailing environmental conditions.

The flexible hierarchical control architecture of HAECN 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 HAECN in a stabilized network environment. It consistently achieves high throughput and sig-

nificantly low latency, thereby ensuring stable and efficient network operations. These results indicate that HAECN presents a promising solution for optimizing network performance in stable data center networks, surpassing ACC in terms of stability, congestion control, and overall performance.

6 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 [14,33], DCQCN, and their enhanced iterations [16], 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* [17], 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 [18], Indigo [19], Vivace [20], and Aurora [21] 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.

7 CONCLUSION

This paper presents HAECN, a novel methodology for addressing congestion control and determining optimal ECN thresholds DCNs. Unlike existing approaches, HAECN leverages DRL techniques to dynamically adapt ECN thresholds based on the evaluation conducted by its decision modules. A key contribution of HAECN 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 stable environments. Experimental evaluations demonstrate the superior performance of HAECN compared to existing methods, showcasing its ability to significantly reduce computational overhead and enhance network functionality. Specifically, HAECN 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.

References

1. Chen, T., Li, M., Li, Y., Lin, M., Wang, N., & Wang, M., et al.: Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. In: arXiv preprint arXiv:1512.01274. (2015)
2. Bunnag, C., Jareoncharsri, P., Tantilipikorn, P., Vichyanond, P., & Pawankar, R.: Epidemiology and current status of allergic rhinitis and asthma in Thailand–ARIA Asia-Pacific Workshop report. In: Asian Pac J Allergy Immunol, 27(1), 79-86.(2009)
3. Lu, X., Islam, N. S., Wasi-Ur-Rahman, M., Jose, J., Subramoni, H., Wang, H., & Panda, D. K.: High-performance design of Hadoop RPC with RDMA over InfiniBand. In: 2013 42nd International Conference on Parallel Processing pp. 641-650. IEEE. (2013)
4. Ramakrishnan, K., Floyd, S., & Black, D.: The addition of explicit congestion notification (ECN) to IP. In: No. rfc3168. (2001)
5. Yan, S., Wang, X., Zheng, X., Xia, Y., Liu, D., & Deng, W.: ACC: Automatic ECN tuning for high-speed datacenter networks. In: Proceedings of the 2021 ACM SIGCOMM 2021 Conference pp. 384-397. (2021)
6. Abbasloo, S., Yen, C. Y., & Chao, H. J.: Classic meets modern: A pragmatic learning-based congestion control for the internet. In: Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication pp. 632-647. (2020)
7. Tian, H., Liao, X., Zeng, C., Zhang, J., & Chen, K.: Spine: an efficient DRL-based congestion control with ultra-low overhead. In: Proceedings of the 18th International Conference on emerging Networking EXperiments and Technologies pp. 261-275. (2022)
8. Li, Y., Alizadeh, M., Yu, M., Miao, R., & Kelly, F.: HPCC: High precision congestion control. In: Proceedings of the ACM Special Interest Group on Data Communication pp. 44-58. (2019)

9. Mittal, R., Lam, V. T., Dukkipati, N., Blem, E., Wassel, H., & Ghobadi, M., et al.: TIMELY: RTT-based congestion control for the datacenter. In: ACM SIGCOMM Computer Communication Review, 45(4), 537-550. (2015)
10. Zhu, Y., Eran, H., Firestone, D., Guo, C., Lipshteyn, M., & Liron, Y., et al. In: ACM SIGCOMM Computer Communication Review, 45(4), 523-536.(2015)
11. Network Simulator, `https://wwwnsnam.org`. April 2023
12. Paszke A, Gross S, Massa F, et al. : Pytorch: An imperative style, high-performance deep learning library. In: Advances in neural information processing systems, 32. (2019)
13. Yin, H., Liu, P., Liu, K., Cao, L., Zhang, L., Gao, Y., & Hei, X.: ns3-ai: Fostering artificial intelligence algorithms for networking research. In: Proceedings of the 2020 Workshop on ns-3, pp. 57-64. (2020)
14. Alizadeh, M., Greenberg, A., Maltz, D. A., Padhye, J., Patel, P., Prabhakar, B., & A., et al.: Data center tcp (dctcp). In: Proceedings of the ACM SIGCOMM 2010 Conference pp. 63-74.(2010)
15. Alizadeh, M., Javanmard, A., & Prabhakar, B.: Analysis of DCTCP: stability, convergence, and fairness. In: ACM SIGMETRICS Performance Evaluation Review, 39(1), 73-84. (2011)
16. Radhika Mittal, Alexander Shpiner, Aurojit Panda, Eitan Zahavi, Arvind Krishnamurthy, Sylvia Ratnasamy, and Scott Shenker: Revisiting Network Support for RDMA. In: ACM SIGCOMM. (2018)
17. Wu, H., Ju, J., Lu, G., Guo, C., Xiong, Y., & Zhang, Y.: Tuning ECN for data center networks. In: Proceedings of the 8th international conference on Emerging networking experiments and technologies, pp. 25-36. (2012)
18. Winstein, K., & Balakrishnan, H.: Tcp ex machina: Computer-generated congestion control. In: ACM SIGCOMM Computer Communication Review, 43(4), 123-134. (2013)
19. Yan, F. Y., Ma, J., Hill, G. D., Raghavan, D., Wahby, R. S., Levis, P., & Winstein, K.: Pantheon: the training ground for Internet congestion-control research. In: 2018 USENIX Annual Technical Conference (USENIXATC 18), pp. 731-743. (2018)
20. Dong, M., Meng, T., Zarchy, D., Arslan, E., Gilad, Y., Godfrey, B., & Schapira, M.: PCC vivace: Online-learning congestion control. In: 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18), pp. 343-356. (2018)
21. Jay, N., Rotman, N., Godfrey, B., Schapira, M., & Tamar, A.: A deep reinforcement learning perspective on internet congestion control. In: International Conference on Machine Learning, pp. 3050-3059. PMLR. (2019)
22. Xu, R., Li, W., Li, K., Zhou, X., Qi, H.: DarkTE: Towards Dark Traffic Engineering in Data Center Networks with Ensemble Learning. In: Proceedings of IEEE/ACM IWQOS, pp. 1-10 (2021)
23. Liu, Y., Li, W., Qu, W., Qi, H.: BULB: Lightweight and Automated Load Balancing for Fast Datacenter Networks. In: Proceedings of ACM ICPP, pp. 1-11 (2022)
24. Li, W., Yuan, X., Li, K., Qi, H., Zhou, X.: Leveraging endpoint flexibility when scheduling coflows across geo-distributed datacenters. In: Proceedings of IEEE INFOCOM, pp. 873-881 (2018)
25. Li, W., Chen, S., Li, K., Qi, H., Xu, R., Zhang, S.: Efficient online scheduling for coflow-aware machine learning clusters. IEEE Transactions on Cloud Computing, 10(4), pp. 2564-2579 (2020)
26. He, X., Li, W., Zhang, S., Li, K.: Efficient Control of Unscheduled Packets for Credit-based Proactive Transport. In: Proceedings of ICPADS, pp. 593-600 (2023)

27. Wang, J., Rao, S., Ying, L., Sharman, P.K., Hu, J.: Load Balancing for Heterogeneous Traffic in Datacenter Networks. Journal of Network and Computer Applications, Vol. 217, (2023)
28. Hu, J., Huang, J., Li, Z., Wang, J., He, T.: A Receiver-Driven Transport Protocol with High Link Utilization Using Anti-ECN Marking in Data Center Networks. IEEE Transactions on Network and Service Management, 20(2), pp. 1898-1812 (2022)
29. Hu, J., Zeng, C.,Wang, Z., Zhang, J., Guo, k., Xu, H., Huang, J., chen, k.: Enabling Load Balancing for Lossless Datacenters. In Proc. IEEE ICNP, (2023)
30. Hu, J., He, Y., Wang, J., Luo, W., Huang. J.: RLB:Reordering-Robust Load Balancing in Lossless Datacenter Network. In Proc. ACM ICPP, (2023)
31. Hu, J., Zeng, C., Wang, Z., Xu, H., Huang, J., Chen, K.: Load Balancing in PFC-Enabled Datacenter Networks. In: Proceedings of ACM APNet (2022)
32. Floyd, S., & Jacobson, V.: Random early detection gateways for congestion avoidance. In: IEEE/ACM Transactions on networking, 1(4), 397-413. (1993)
33. Cardwell, N., Cheng, Y., Gunn, C. S., Yeganeh, S. H., & Jacobson, V.: BBR: congestion-based congestion control. In: Communications of the ACM, 60(2), 58-66. (2017)
34. Chung, J., Ahn, S., & Bengio, Y.: Hierarchical multiscale recurrent neural networks. In: arXiv preprint arXiv:1609.01704. (2016)
35. Gawowicz, P., & Zubow, A.: ns3-gym: Extending openai gym for networking research. In: arXiv preprint arXiv:1810.03943. (2018)
36. Abbasloo, S., Yen, C. Y., & Chao, H. J.: Wanna make your TCP scheme great for cellular networks? Let machines do it for you!. In: IEEE Journal on Selected Areas in Communications, 39(1), 265-279. (2020)
37. Wang, J., Yuan, D., Luo, W., Rao, S., Sherratt, R.S., Hu, J.: Congestion Control Using In-Network Telemetry for Lossless Datacenters. CMC-Computer, Materials and Continua, 75(1), 1195-1212 (2023)
38. Wang, J., Liu, Y., Rao, S., Sherratt, R.S., Hu, J.: Enhancing Security by Using GIFT and ECC Encryption Method in Multi-tenant Datacenters. CMC-Computer, Materials and Continua, 75(2), 3849-3865 (2023)
39. Hu, C., Liu, B., Zhao, H.: DISCO: Memory Efficient and Accurate Flow Statistics for Network Measurement. In Proc. IEEE ICDCS, pp. 665-674 (2010)
40. Li, H., Zhang, Y., Zhang, Z.: Ursa: Hybrid Block Storage for Cloud-Scale Virtual Disks. In Proc. ACM EuroSys, pp. 1-17 (2019)
41. Bai, W., Chen, K., Hu, S., Tan, K., Xiong, Y.: Congestion Control for High-speed Extremely Shallow buffered Datacenter Networks. In Proc. ACM APNet, pp.29-35 (2017)
42. Wang, Y., Wang, W., Liu, D., Jin, X., Jiang, J., Chen, K.:Enabling edge-cloud video analytics for robotics applications.In: Proceedings of IEEE Transactions on Cloud Computing, (2022)
43. Li, Z., Bai, W., Chen, K.: Rate-aware flow scheduling for commodity data center-networks. In Proc. IEEE INFOCOM, pp. 1-9 (2017)
44. Zhao, Y., Huang, Y., Chen, K.: Joint VM placement and topology optimization for traffic scalability in dynamic datacenter networks. Computer Networks, pp. 109-123, (2015)
45. Hu, C., Liu, B., Zhao, H.: Discount counting for fast flow statistics on flow size and flow volume. IEEE/ACM Transactions on Networking, 22 (3), 970-981, (2014)
46. Zheng, J., Du, Z., Zha, Z., Yang, Z., Gao, X., Chen, G.: Learning to Configure Converters in Hybrid Switching Data Center Networks. IEEE/ACM Transactions on Networking, 1-15 (2023)