Deep Reinforcement Learning based Load Balancing for Heterogeneous Traffic in Datacenter Networks*

Jinbin Hu, Wangqing Luo, Yi He, Jing Wang, and Dengyong Zhang

School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410004, China
{jinbinhu,znwj_cs,zhdy}@csust.edu.cn
{luowangqing,heyi}@stu.csust.edu.cn

Abstract. Modern high-speed datacenter networks (DCNs) employ multi-tree topologies to provide large bisection bandwidth. Load balancing is crucial for making full use of parallel equal-cost paths and ensuring high link utilization. In the past decades, a large number of heuristic load balancing mechanisms have been proposed to alleviate congestion. However, they cannot be resilient to the concurrent explosive growth and unpredictable dynamic traffic scenarios. Recently, deep reinforcement learning methods have become very powerful techniques for reacting to dynamically changing networks, but the existing learning-based load balancing schemes are agnostic to the heterogeneous traffic generated by diverse applications. Thus, the latency-sensitive short flows suffer from large tail delays due to the long pre-learning period. To address the above issues, we propose a deep reinforcement learning-based load balancing mechanism called DRLB. Specifically, DRLB differentiates heterogeneous traffic by using Deep Reinforcement Learning (DRL) in conjunction with the Distributed Distributional Deterministic Policy Gradients (D4PG) algorithm to make the optimal (re)routing for long flows while adopting the Weighted Cost Multipathing (WCMP) mechanism for short flows. The NS-3 experimental results show that, DRLB increases the throughput of long flows by up to 47% and reduces the flow completion time (FCT) of short flows by up to 58% compared to the state-of-the-art load balancing mechanisms.

Keywords: Datacenter Networks · Deep Reinforcement Learning · Load Balancing · Heterogeneous Traffic

* Dengyong Zhang 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).

1 Introduction

In recent years, as the stringent demands of high throughput and low latency for applications like cloud computing, big data, and artificial intelligence have continued to expand, modern datacenter networks have been gradually scaling up with rising bandwidth and traffic [1] [2] [3]. As a result, the dynamic network environment and real-time traffic changes have become more pronounced [4] [5] [6]. For instance, during each round of model training iterations in the present large-scale distributed training, the parameter synchronization updating procedure generates a significant amount of traffic interactions at the network endpoints [7].

To support large bisection bandwidth, production datacenter networks usually employ multi-rooted tree architectures. Existing load balancing mechanisms, however, confront emerging challenges in utilizing these equivalent multipaths, enhancing link utilization, and decreasing Flow Completion Time (FCT) to satisfy the requirements of varied applications [8]. As a result, developing advanced load balancing mechanisms has become an important research direction for datacenter networks [9].

Recently, a number of load balancing mechanisms have been proposed, but they aren't performing effectively in this constantly shifting environment. In particular, the inability to switch paths flexibly in ECMP and Freeway [10] [11], two flow-based scheduling granularity mechanisms, results in ineffective link utilization. Two packet-based scheduling granularity mechanisms, DRILL and RPS [12], might worsen overall performance by increasing the frequency of disorder in dynamic network scenarios. Hermes and TLB mechanisms also attempt to tackle the challenge by distinguishing heterogeneous traffic and utilizing routes with various granularities [13] [14], but they also struggle to adapt to the dynamically changing network environment.

The problem of load balancing offers novel solutions due to the development of artificial intelligence techniques [15] [16] [17]. Deep reinforcement learning is used in datacenter networks to continuously interact with the network environment and learn the best load balancing strategy in order to adapt to the dynamically changing network environment and increase network performance and reliability [18] [19].

We propose a heterogeneous traffic load balancing mechanism (DRLB) based on Deep Reinforcement Learning (DRL) to tackle the aforementioned problem. This mechanism distinguishes heterogeneous traffic by employing DRL for long flows and a straightforward Weighted Cost Multipathing (WCMP) mechanism for short flows [20]. It is important to note that in this paper, the long flows to be processed by us are accumulated according to the sending rate over the cycle time of the host-side parameter updates. We build a network model applicable to DRLB mechanism that can observe the network state in real time as input to the Distributed Distributional Deterministic Policy Gradient (D4PG) algorithm [21] and dynamically adjust the Link State Metric (LSM) to guide the forwarding of long flows through DRL agent, where a lower LSM value indicates a better link. By distinguishing heterogeneous traffic, we ensure optimal routing for all

varieties of traffic, ensuring low latency for short flows and high throughput for long flows. The main contributions of this paper are as follows:

- We highlight the inadequacies and issues of the load balancing mechanisms presently employed in datacenters to deal with network changes and congestion. We devise a novel load balancing mechanism by combining machine learning techniques and investigate the problems of DRL pre-learning and short flow mismatch in order to address these issues. Experiments verify the rationality and scientific validity of these problems.
- We propose a deep reinforcement learning-based heterogeneous traffic load balancing mechanism called DRLB that applies DRL on long flows and a WCMP mechanism on short flows. DRLB reduces the 99^{th} percentile FCT of short flows and increases the throughput of long flows. We develop a network model applicable to DRLB mechanism that monitors the network state in real time as the input to D4PG algorithm and dynamically modifies LSM via DRL agent to direct the forwarding of long flows. In addition, a plausible state action space and reward function are proposed.
- We conduct a series of experiments on the NS-3 simulator involving diverse traffic workloads and symmetric and asymmetric topology scenarios. We compared the experimental results to the existing load-balancing mechanisms, including Freeway, Hermes, and TLB. The experimental results demonstrate that DRLB substantially increases the throughput of long flows and decreases the FCT of short flows at wide-range traffic load.

The rest of this paper is organized as following. In Section 2, we discuss the context and rationale for this paper. Section 3 describes DRLB mechanism's design and implementation in detail. Section 4 provides an experimental evaluation of DRLB mechanism and a comparison to other mechanisms. Section 5 examines load balancing mechanisms in datacenter networks and DRL-related research. In Section 6, we summarize the paper's contributions, key findings, and prospective research directions.

2 Background and Motivation

2.1 Background

Load balancing. In recent years, numerous load balancing mechanisms have been proposed to enhance the performance of datacenter networks. When designing these mechanisms, numerous considerations must be made, such as selecting the appropriate granularity for load balancing. A smaller granularity offers more routing options, enhancing network utilization and load balancing performance. Granularity can be packet (DRILL mechanism) [12], flow (WCMP mechanism) [20], flowlet (CONGA mechanism) , or adaptive (TLB mechanism) [14] among the existing mechanisms [22] [23] [24]. In this paper, DRLB mechanism handles heterogeneous traffic differently, with short flows at the granularity of flows and

long flows at the granularity of adaptive flows formed by the host-side sending rate accumulated over DRL agent iteration time. In addition to granularity, network congestion-aware policies, mechanism control structure, deployment difficulty, and scalability must be considered when devising a load balancing mechanism [25] [26] .

However, traditional load balancing mechanisms continue to encounter numerous challenges in the face of growing datacenter network size and complexity. Traditional mechanisms are incapable of adapting to the heterogeneous devices and complex topologies encountered in datacenter networks, nor can they precisely assess network state and traffic distribution [27] [28] [29]. In addition, highly unpredictable and bursty traffic in datacenter networks frequently causes network failures, while traditional load balancing mechanisms are unable to modify their policies in a timely manner, exacerbating the issues of congestion and performance degradation. This paper proposes DRLB mechanism, which combines deep reinforcement learning techniques to adaptively adjust the load balancing policy by learning the characteristics of network state and traffic distribution, which can better adapt to the dynamic network environment and improve the performance and reliability of the data center network.

Deep reinforcement learning. DRL has emerged as a significant branch of artificial intelligence technology in recent years, achieving remarkable results in natural language processing, computer vision, speech recognition, and other fields. DRL combines the information extraction ability of deep learning models with the decision-making ability of reinforcement learning to achieve abstraction and feature extraction of complex data through multi-layer nonlinear mapping and to provide excellent solutions for complex tasks driven by the reinforcement learning objective [30] [31].

DRL has robust generalization and adaptability to meet the challenges of complex environments and large-scale tasks, which is one of the reasons why we are combining it with datacenter network problems [32] [33]. DRL excels at coping with high-dimensional, complex environments, extracting high-level features from raw input data, and modeling the environment effectively. Moreover, DRL is able to learn complex policies and performs exceptionally well when confronted with complex tasks. Due to its ability to train end-to-end, it can effectively address difficult-to-model problems in datacenter networks, such as complex and variable traffic and enormous scope .

However, DRL is not entirely devoid of disadvantages. Its training requires vast quantities of data and computational resources, a lengthy training period, and an unstable training process that requires careful calibration of the network structure and hyperparameters to assure convergence and stability. In order to solve the problem of network load balancing in a datacenter, the appropriate methods and algorithms must be chosen based on the specific mission requirements and constraints [34] [35]. This paper's DRLB mechanism utilizes D4PG algorithm, which combines the Deep Deterministic Policy Gradient (DDPG) algorithm and distributed experience replay technology to efficiently utilize large-

scale data for accurate and stable policy learning. Dynamically, D4PG algorithm is able to intelligently and dynamically modify the load balancing policy to adapt to network environment changes as well as diverse traffic distribution and device heterogeneity [21]. By applying D4PG algorithm to the design of load balancing mechanisms, network performance and dependability can be enhanced while maintenance costs and the need for manual intervention are reduced.

2.2 Motivation

In this subsection, we experimentally validate the load balancing mechanisms described in the introduction and the learning period problem for DRL.

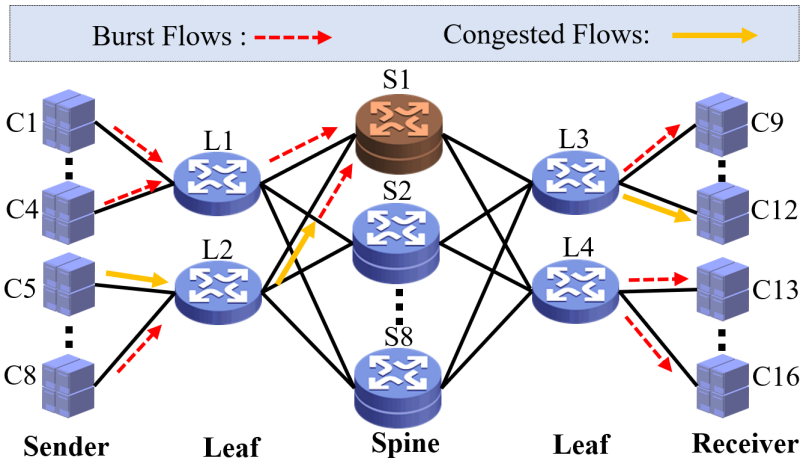


Fig. 1. Leaf-Spine topology

Load balancing problems. Using the NS-3 simulation platform, we conduct experiments to address the problems of existing load balancing mechanisms (Freeway, Hermers, and TLB) in adapting to changing network environments. In the leaf spine topology shown in Fig.1, the sender C_5 transmits numerous data flows to the receiver C_{12}. According to the experimental findings, the spine switch S_1 is overloaded by a large number of bursty long flows when the three load balancing mechanisms tend to transmit smoothly and the throughput is largely stable. Specifically, Fig. 2 demonstrates that the long flow throughput of the three load-balancing mechanisms increases by approximately 35% at approximately 30 ms. We offer a comprehensive analysis of these three load balancing mechanisms.

First, for TLB mechanism, which performs best in Fig. 2, it can only sense local congestion information and cannot acquire entirely accurate, real-time, and comprehensive congestion information. In the case that S_1 becomes congested

due to a large number of burst long flows, L_2 is unable to detect this and may continue to allocate traffic to S_1, resulting in a performance degradation. Although Hermes can detect global congestion and understand that a large number of congested long flows occur in S_1, its overly conservative rerouting mechanism prevents it from resolving the S_1 congestion problem in a timely manner, resulting in throughput increases but actual throughput decreases. Although Freeway mechanism senses the change in path congestion and detects the congestion problem in S_1, its algorithm to reclassify paths to select high-throughput paths is not timely, and the network congestion may intensify during this period and new congested links may appear, resulting in poor overall performance.

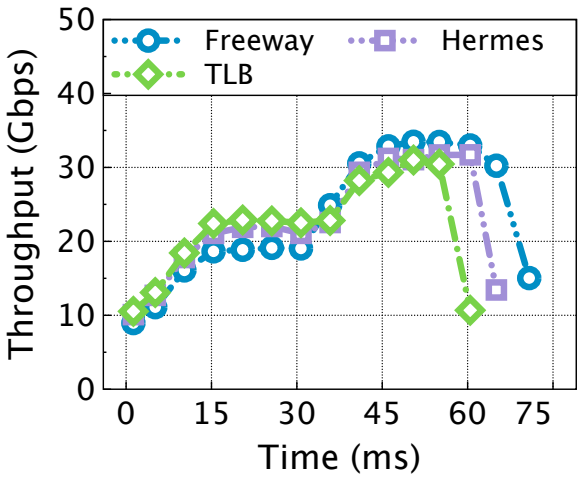


Fig. 2. Throughput of long flows under three load balancing mechanisms

Deep reinforcement learning problems. The fact that the learning period of DRL is too lengthy for delay-sensitive short flows is one of the challenges associated with the application of DRL algorithms in the field of load balancing. Since DRL algorithms need to interact with the environment in order to learn the optimal policy, it takes time for these interactions to accumulate sufficient experience and optimise the policy. However, delay-sensitive short flows have stringent requirements for timeliness and need to be communicated quickly, whereas the learning process of DRL algorithms is lengthy and may not be able to accommodate the short delay in time. When the short flows arrive, the algorithm may still be in the process of learning and cannot instantaneously assign the optimal action.

To demonstrate our point, we conduct the DPG and DDPG algorithms using TensorFlow and PyTorch, respectively, as machine learning frameworks. In order to devise an appropriate neural network model, we created a multilayer perceptron network with two hidden layers, each containing 256 neurons. Such a

network structure can provide sufficient expressive capacity for illustrating and learning complex load balancing policies in datacenter networks. To simulate real-world traffic conditions, we set up short flows that are sensitive to delay and large-scale long flows. The average sending rate for short flows is set to 100 Mbps, and the traffic size is generated randomly between 10 KB and 100 KB. The average sending rate for long flows is 1 Gbps, and the generated traffic size is randomly distributed between 1 MB and 10 MB. This type of traffic is typically very sensitive to latency, necessitating quick and precise load balancing mechanisms to ensure timely processing and delivery.

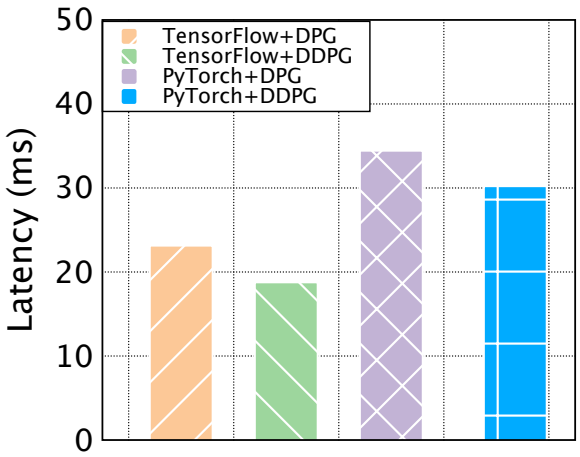


Fig. 3. Latency of two algorithms in different frameworks

Based on the experimental results depicted in Fig. 3, we observe a minimum processing latency of 23 ms when conducting the most advanced machine learning algorithms on both frameworks. This indicates that the short flows have already completed transmission before DRL algorithm implements the load-balancing guidance action. Under the current configuration, this result indicates that DRL algorithm has some latency issues when processing delay-sensitive short flows. Since the transmission time of short flows is relatively short, DRL algorithm requires some time to sense the network state, make decisions, and perform load balancing actions, resulting in some short flows having completed transmission before the algorithm becomes active.

3 DRLB Design

3.1 Design overview

In this section, we briefly describe DRLB mechanism's guiding principles. Each server in the leaf-spine topology, as depicted in Fig. 4, contains a DRL agent

responsible for network data collection, data processing, feature selection, model training, and output command. On top of each server, DRL algorithm module deploys a distributed actor-critic. Using WCMP mechanism, the actor obtains real-time network data (e.g., throughput, link utilization, and latency) from the switch and dispatches short flows to the link with the lowest link weight.

For long flows, the distributed actor stores its generated experience in the replay buffer, while the critic samples the replay buffer and modifies the priority. Then, based on the sample information, the critic assesses the performance and load of the links and outputs LSM values via the learner to direct the distributed actor to choose the optimal link for forwarding long flows. Utilizing the reinforcement learning capability of D4PG algorithm, the forwarding policy of extended flows is dynamically amended based on the real-time network state and traffic characteristics, resulting in improved load balancing.

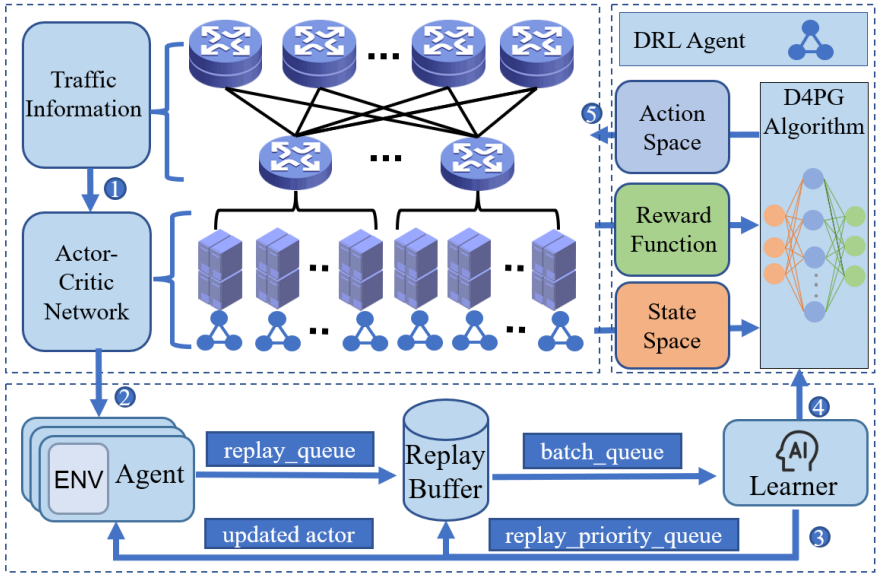


Fig. 4. DRLB overview

DRL agent is composed of four components:

- **State module.** A suitable state representation is devised in order for DRL algorithm to comprehend the current state of the datacenter network. This module aggregates real-time network information from the switches, including throughput, link utilisation, and latency, for use as input to DRL in order to inform the agent's decisions.
- **Action module.** The decision made by DRL agent based on the current condition is an action. In the problem of load balancing, the action con-

sists of selecting the optimal link for forwarding long flows. Since the action space is continuous, we employ D4PG algorithm to address the difficulty of continuous action space.

- **Reward module.** Reward is the feedback an agent receives based on their actions. In the load balancing problem, we construct an appropriate reward function based on network performance metrics such as latency to help agents learn the correct load balancing policy. The objective is to use rewards to encourage agents to learn load-balancing policies that can enhance network performance and reliability. An appropriate reward structure can motivate agents to take measures to enhance network performance.
- **DRL algorithm module.** By learning and optimizing their policies, the algorithm is used to train agents to progressively improve their performance on load balancing problems. To solve the load balancing problem, we employ D4PG algorithm, which can handle the challenges of continuous action space and high-dimensional state space, learn the optimal load balancing policy, and maximize the cumulative reward by optimizing the policy network's parameters.

3.2 Design detail

State module. The state module is a crucial component of DRL, used to characterize the present state of the environment and provide a foundation for intelligent agents' decision-making and learning.

Due to the enormous amounts of traffic and variable and bursty states, the original state representation in a datacenter network environment may be too large or challenging to manage. Therefore, we reduce the dimension of the state space by selecting only the significant characteristics that have an effect on the load balancing issue. These characteristics include network traffic, link utilization, latency, and bandwidth data. On the basis of the specific load balancing targets and requirements, essential network state-reflecting characteristics are chosen to avoid accumulating redundant or irrelevant data. The data acquired by the state module cannot be inputted directly and must be processed and transformed into a format that is compatible with DRL model before it can be inputted.

In this paper, these data are transformed into a vector matrix. This paper describes the deployment of DRLB mechanism on an $N*M$ leaf spine topology with K servers connected to each leaf switch. We collect queue length (QL), throughput (T), link utilization (LU), and latency (L) information at the switches and design them as state vectors for DRLB:

$$\boldsymbol{S} = \begin{bmatrix} S_N \\ S_M \end{bmatrix} \tag{1}$$

$$\boldsymbol{S}_N = \begin{bmatrix} S_{N_1} \\ S_{N_2} \\ \cdots \\ S_{N_N} \end{bmatrix} = \begin{bmatrix} QL_{N1}, T_{N_1}, LU_{N_1}, L_{N_1} \\ QL_{N2}, T_{N_2}, LU_{N_2}, L_{N_2} \\ \cdots \\ QL_{N_N}, T_{N_N}, LU_{N_N}, L_{N_N} \end{bmatrix} \tag{2}$$

where $S_{Ni} = [QL_{Ni}, T_{Ni}, LU_{Ni}, L_{Ni}]$ represents the real-time queue length and throughput information collected on the $i-th$ leaf switch. The vectors S_M and S_N on the M spine switches are obtained identically, so we will not list them individually here.

Action module. Another essential component of DRL is the action module, which defines the set of actions that an agent can choose from. When designing the action module in DRLB mechanism, the agent-selectable actions are defined by inputting the calculated state into DRL agent in accordance with the load balancing objectives and requirements of DRLB mechanism. The action module transmits LSM values to the host side, instructing it to forward long flows to the finest integrated link. The action module defines each action's meaning and operation explicitly, allowing the agent to select the action with the highest reward value. Since the actions for load balancing are continuous, DRL agent needs to select a continuous set of parameter values to execute the action operations. In this instance, the action space is defined as a contiguous parameter space, and the agent chooses the optimal action to accomplish load balancing through LSM.

Similarly to the state module, the action space module should be able to be dynamically updated to accommodate environmental changes. In a datacenter network environment, the network state and traffic distribution may change, so the action space module has to be able to reflect these alterations and promptly update the collection of optional actions. By appropriately designing and optimizing the action space module, DRL agent can direct the host side to achieve the load balancing goal based on the current state information and the output LSM value and continuously learn and improve its decision strategy to enhance the performance and effectiveness of the system, as described below for DRLB action space:

$$\boldsymbol{A} = [A_1, A_2, \ldots, A_K] \tag{3}$$

where A_i indicates that the i_{th} host end at this moment is guided by LSM value and the current state to select the action with the greatest reward value, thereby optimizing the transmission link for the long flow.

Reward module. DRLB's reward function incorporates the round-trip latency (RTT) of all links and calculates the total reward value by aggregating them. This enables the case of all equivalent links on the leaf-spine topology to be considered, as opposed to only individual links. By comparing the RTT of each link to the average RTT, the reward function can determine the congestion level on the link. When a link's RTT exceeds the average RTT, the reward value becomes negative, indicating a higher level of congestion. The reward function that includes a logarithmic transformation can restrict the reward value to an appropriate range. The logarithmic function can make the change in reward value more gradual and prevent extreme values. Also, the average RTT is used as a reference value in the reward function so that the reward function can adapt

to changes in the network and thus guide DRL agent to make adjustments and decisions accordingly. DRLB reward function is depicted by Equation (4):

$$R = \frac{1}{N \times M} \log \left(\sum_{i=1}^{N \times M} \left(-\frac{RTT_i - \overline{RTT}}{N \times M} \right) \right) \tag{4}$$

N*M represents the total number of equivalent links, where N and M represent the number of leaf and spine switches, respectively. RTT_i is the RTT of the i_{th} link, whereas $\overline{RTT}$ is the average of all connections' RTTs. The reward function is computed by first normalizing and then aggregating the difference between each link's RTT and the average RTT. The value of the normalized difference is divided by $(N \times M)$ to determine the average contribution to all links. The reward value is determined by taking the logarithm of the preceding sum. The condition of the system's load balancing is determined by calculating the difference between the RTT of each link and the average RTT. When all RTTs are close to the average, the value of the reward will be close to zero. And the reward value will become increasingly negative when the RTT of some links deviates significantly from the average. The logarithmic operation of the reward function can limit the incentive value to a narrow range, making gradient optimization and learning simpler to perform.

DRL algorithm module. In this paper, DRLB mechanism selects the policy gradient-based D4PG algorithm. The convergence and stability of D4PG algorithm can accommodate the dynamic and complex nature of the network environment in a datacenter, resulting in significant changes to the reward signal. Using a parameterized policy network, D4PG algorithm generates continuous action values that are appropriate for continuous action selection in load balancing.

In addition, D4PG algorithm has a robust policy search capability; when the network state and load conditions in load balancing problems change frequently, D4PG algorithm is able to modify the policy based on real-time environment information to accommodate varying load demands. D4PG algorithm's replay caching mechanism permits sampling and utilization of historical experience during training. This is crucial for the load balancing problem because it enables DRL agent to learn the policy under a variety of network states and load conditions, thereby enhancing overall performance and adaptability.

The objective of DRLB mechanism is to optimize the performance of the datacenter network, including throughput, latency, and resource utilization metrics.D4PG algorithm discovers policies by maximizing reward value, which can be optimized for these performance indices. Through proper design of the reward function and optimization of the policy network, D4PG algorithm can help DRLB mechanism attain improved network performance.

D4PG algorithm is selected as DRL algorithm for DRLB mechanism. When using D4PG algorithm for load balancing design, it is also necessary to consider network topology, state modeling, reward function design, policy network ar-

chitecture, and training parameter optimization for improved performance and results. Table 1 displays the primary parameters involved in D4PG algorithm.

Table 1: Main parameters involved in D4PG algorithm.

Parameters	Meaning of parameters
$s \in S, a \in A, r \in R$	States, Actions, Rewards
$(s_{i:i+V}, a_{i:i+V-1}, r_{i:i+V-1})$	Status , Action, Reward Sequence
P_i	Sampling priority
D	Replay buffer size
W	The number of distributed actors
V	Sample time length
U	Sample size
ς	Degree of agent exploration
ρ_0 and ϱ_0	Initial learning rate value
γ	Reward discount factor
Z_μ	Random variables
d	Difference between value and target value distribution

Here, we parse D4PG algorithm of DRLB, initializing the network parameters (θ, μ) and designating them to the target network (θ', μ'), as well as some hyperparameters, and and deploy W actors by duplicating the network weights for each actor. (lines 1-4). During the training process, the policy network is kept up-to-date by making training blocks using this sampling method.Then, by constructing the target distribution, we can obtain the target Q values for each step, compare them with the predicted values of the value function network, and calculate the loss function of the value function network.

Finally, by calculating the updates of actors and critics, we can update the parameters of the strategy network and the value function network. In lines 10 and 11, we update the target network and propagate the network weights to the actors to ensure that the target network is periodically updated and that the actor and learner weights are in sync for an efficient DRL training process. When the predetermined number of training cycles have been completed, the current policy parameter is returned as the final policy parameter.

In this paper, LSM value is introduced to characterize the advantages and disadvantages of the current real-time link, and DRL agent outputs LSM value to assist the host side in performing the most reasonable load balancing operation on the long flow. Since D4PG needs to maximize LSM value through extensive learning, we have to improve D4PG algorithm's exploration and speed up its training by presetting an optimal action based on LSM value, which we refer to as alsm. As shown in line 15 of the code, $p_{(t)}$ is a decreasing function; as t increases, the effect of the a_{lsm} value on the agent's choice of action diminishes, and the policy network's action is more likely to be chosen. DRLB executes the action and interacts with the environment based on the selected action in the subsequent phase. After the action is performed, the reward r and the subsequent state s'

are observed. The experience replay buffer maintains a tuple with the current state s, action a, reward r, and next state s. Past experiences are stored and sampled in the experience replay buffer for optimal learning. D4PG algorithm for DRLB is described below:

Algorithm 1: DRLB's D4PG algorithm

Initialization stage:
Hyper-parameters: D, W, V, U, ς, ρ_0 and ϱ_0
Random network weights(θ, μ),target weights $(\theta', \mu') \leftarrow (\theta, \mu)$
Initiate W actors while replicating (θ, μ) to each actors.
Learning Stage:
for t = *1,...,T* **do**
Sample U transitions $(s_{i:i+V}, a_{i:i+V-1}, r_{i:i+V-1})$ of length V with priority P_i from replay buffer D
Target distributions: $Y_i = \sum_{v=0}^{V-1} \gamma^v r_{i+V} + \gamma^V Z_{\mu'}(s_{i+V}, \pi_{\theta'}(s_{i+V}))$
Update the actors and critics:
$\delta_\mu = \frac{1}{U} \sum_i \nabla_\mu (Dp_i)^{-1} d(Y_i, Z_\mu(s_i, a_i))$
$\delta_\theta = \frac{1}{U} \sum_i \nabla_\theta \pi_\theta (\mathbf{s}_i) E [\nabla_{\mathbf{a}} Z_\mu (\mathbf{s}_i, \mathbf{a})]\big|_{\mathbf{a}=\pi_\theta(\mathbf{s}_i)}$
Update network parameters:$\theta \leftarrow \theta + \rho_t \delta_\theta, \mu \leftarrow \mu + \varrho_t \delta_\mu$
If $t = 0$ mod t_{target}, updating target network $(\theta', \mu') \leftarrow (\theta, \mu)$
If $t = 0$ mod t_{actors}, replication of network weights to actors
end
Return θ
Actor

Repeat
Sample random action $a = (1 - p(t)) * \pi_\theta(s) + \varsigma V(0, 1) + p(t) * a_{lsm}$
Execute Sample action a, then observe reward r and the next state s'
Store (s, a, r, s') in replay buffer
Until learner completes

Through distributed learning, distribution-based value function estimation, distributed empirical playback, prioritized empirical playback, and model-free policy optimization, DRL algorithm of DRLB provides an efficient, stable, and accurate method for solving the load balancing problem in continuous action space.

4 Experimental Results and Evaluation

4.1 Experimental Setup

In this subsection, using the NS-3 simulator, we conduct a series of large-scale experiments to evaluate the performance of DRLB mechanism under various traffic and topology scenarios. We compare DRLB mechanism to several existing load-balancing algorithms (Freeway, Hermers, and TLB). The 99^{th} percentile

FCT of short flows and the throughput of long flows are choosed as the evaluation metric for a comprehensive performance evaluation of each load balancing algorithm.Table 2 displays the remaining experimental settings:

Table 2: Key parameters of DRLB in experiments.

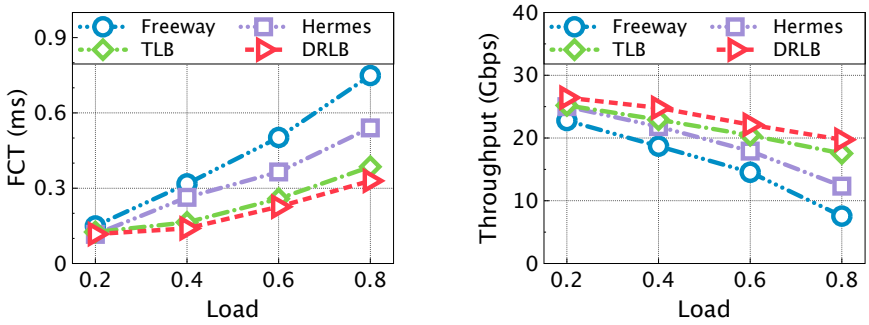
Parameters	Value/Description
Number of leaf switches	8
Number of spine switches	12
Number of end-hosts	40
Number of links	1
Spine-Leaf Capacity	40Gbps
Leaf-Server Capacity	40Gbps
Link Latency	2us
Realistic datacenter workloads	Web Sever and Data Mining Communication
Pattern	All-to-all
Transport	DCTCP
Switch buffer	512KB

4.2 Evaluation of Experimental Results

In this paper, we compare the performance evaluation of the Freeway, Hermers, TLB, and DRLB load balancing mechanisms under two actual datacenter workloads, Web server (64KB) and datamining (7.4MB), in a symmetric leaf spine topology. Figs. 5 and 6 demonstrate that as the traffic load increases, the 99^{th} percentile FCT of the short flows increases considerably, whereas the throughput of the long flows decreases as the traffic load increases. Figs. 5(a) and 6(a) demonstrate that DRLB mechanism reduces the FCT of short flows by approximately 54%, 38%, and 12% on average compared to the Freeway, Hermers, and TLB mechanisms, respectively, when the traffic load reaches 80%, and Figs. 5(b) and 6(b) demonstrate that for the throughput of long flows, DRLB mechanism reduces the FCT by approximately 38% compared to the Freeway, Hermers, and TLB mechanisms.

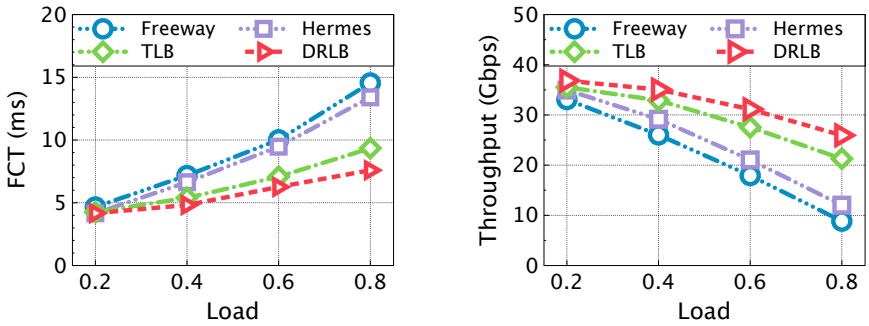
As shown in Fig. 5 and Fig. 6, we can observe that the performance of Hermes mechanism decreases by approximately 30% when the traffic workload increases, and we conclude that this is because Hermes mechanism is not timely enough based on end-system-aware congestion, and the path-aware and re-routing mechanisms are too conservative. In contrast, the performance of DRLB mechanism is much more stable, with a loss of only about 4% in both FCT and throughput on average.

Topologies frequently exhibit asymmetric link congestion in real-world application scenarios due to datacenter network traffic increases, multiple dynamic changes, and other factors. We set the capacity of multiple links in the leaf spine topology on NS-3 to be only 1/5 of other links in order to simulate the asymmetric topology for experiments, and Web server and datamining traffic remain



(a) 99^{th}-ile FCT of short flows (b) Throughput of long flows

Fig. 5. Web server workload under symmetrical topology

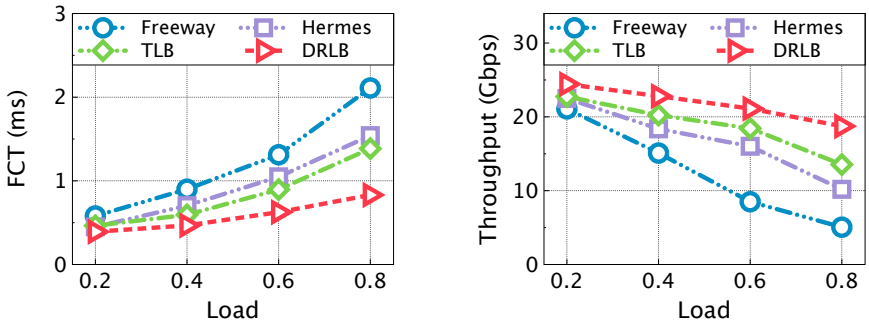


(a) 99^{th}-ile FCT of short flows (b) Throughput of long flows

Fig. 6. Data mining workload under symmetrical topology

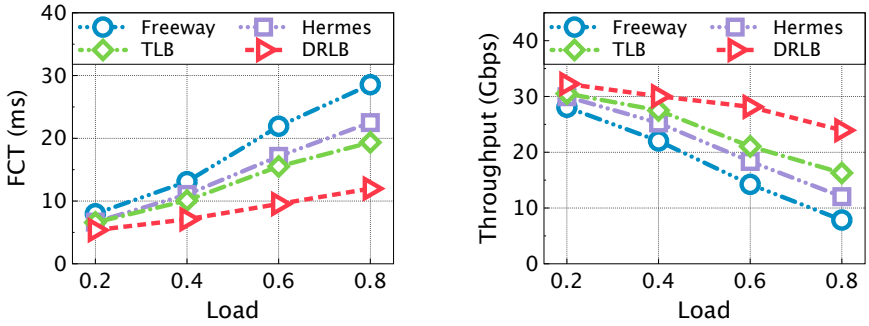
the primary burden. According to Fig. 7(a), DRLB mechanism reduces the 99^{th} percentile FCT of short flows by 63%, 42%, and 29% compared to the Freeway, Hermers, and TLB mechanisms, respectively, under the webserver load. Fig. 8(a) reveals that DRLB mechanism optimizes the performance metric of FCT by more than 28% compared to the other three mechanisms in the datamining scenario. By comparing Fig. 7(b) and Fig. 8(b), we can see that DRLB does a superior job of ensuring high throughput requirements for long flows than Freeway, Hermers, and TLB, where the average throughput of long flows decreases by 49%, 30%, and 25%, respectively.

From the above experimental results, we observe that the asymmetric topology affects the performance of DRLB mechanism by only 3% in both traffic load scenarios, whereas TLB mechanism suffers a performance loss of up to 37%. In the symmetric topology scenario, TLB mechanism performs 27% better than Hermes mechanism under both workloads, but in the asymmetric topology, TLB handles short flows at packet granularity because its mechanism must adaptively



(a) 99^{th}-ile FCT of short flows (b) Throughput of long flows

Fig. 7. Web server workload under asymmetrical topology



(a) 99^{th}-ile FCT of short flows (b) Throughput of long flows

Fig. 8. Data mining workload under asymmetrical topology

adjust the reroute granularity of long flows based on the strength of short flows, while it only senses local congestion and cannot detect the problem in time when congestion occurs in other links. The performance degradation is severe due to the high number of packet retransmissions, which is only about 12% better than Hermes.

By conducting simulation experiments under two traffic loads and two leaf spine topology scenarios to evaluate the two performance metrics, we can conclude that DRLB improves the long flow throughput by 47% and reduces the short FCT by 58% when compared to the existing Freeway, Hermers, and TLB mechanisms. It demonstrates the efficacy and progression of its precise differentiation of heterogeneous traffic into short flows using WCMP mechanism and long flows combined with DRL for load balancing.

5 Related Works

In this paper, we provide a load balancing method for heterogeneous traffic in data center networks based on deep reinforcement learning. We introduce the work associated with this study in two sections in this subsection: load balancing and deep reinforcement learning.

Load Balancing. Based on the various deployment locations, the prevalent load balancing mechanisms can be divided into the following three categories:

- Central controller-based load balancing mechanisms, which primarily include re-routing methods like Hedera ; fine-grained control methods like Fastpass ; the scheduling methods Freeway [11] and AuTO, etc., which differentiate between short and long flows; and the LBDC [36] mechanism , which is used to solve the problem of load imbalance among multiple controllers.
- Host-based load balancing mechanisms include Hermes [13], and other rerouting techniques; Presto, and other traffic slicing protocols; and CAPS [37], and other fine-grained scheduling mechanisms.
- Switch-based load balancing mechanisms need to be distinguished from the scheduling granularity, including flow-level scheduling methods like Dart and WCMP [20]; packet-level scheduling methods like RPS , DRILL [12]; packet-cluster-level scheduling methods like CONGA ; and adaptive granularity scheduling methods like AG [38] and TLB [14].

The central controller-based load balancing mechanism can precisely detect link failures and designate the optimal transmission path for data transfers using global network state information. However, obtaining and maintaining global information necessitates a certain deployment overhead, and the large feedback and control delays hinder load balancing performance under dynamic surge traffic. The host-based load balancing mechanism is more scalable than the centralized scheduling scheme because its hosts are independent and each host can modify the load balancing mechanism to meet its own needs. However, the end-to-end feedback latency of host-based load balancing prevents it from adapting to extremely dynamic surge traffic. The switch-based load balancing mechanism can detect the link load in the network in real time and quickly balance the network traffic on the path connected to the switch out ports, but it is difficult to obtain the end-to-end path accurately and quickly, thereby affecting the accuracy of traffic transfer.

Deep Reinforcement Learning. DRL has become a research hotspot in the field of machine learning in recent years, and DRL algorithms are primarily divided into the following categories:

- Value function-based algorithms employ iterative updates to optimize the value function. These algorithms evaluate the value of each state using the value function, and they update the value function's parameters based on the most recent forecasts and reward signals. Among these, Deep Q-Network (DQN) [19] is a traditional approach that stabilizes training by using experience replay and target networks and a deep neural network to approximate

the value function. By addressing the issue of high valuation functions in DQN to enhance performance, Double DQN (DDQN) improves on DQN.
– Function approximation is used by policy gradient-based algorithms to build policy networks. These algorithms choose actions, retrieve the related reward values via the policy network, and then optimize the policy by maximizing the reward value by adjusting the policy network's parameters in the gradient direction. An actor-based algorithm known as DDPG combines the concepts of value functions with policy gradients. By restricting the size of policy updates, the Proximal Policy Optimization (PPO) [39] algorithm ensures the stability and convergence of policy optimization. These algorithms are built on the concept of policy gradients and enhance the algorithm's performance through various enhancements and optimizations.

In addition, DRL can improve the policy search process by incorporating artificial supervision and integrating the policy evaluation and value estimation capabilities of deep neural networks with the policy search and exploration capabilities of Monte Carlo Tree Search (MCTS) [40]. Hierarchical reinforcement learning methods can also be used to decompose the overall goal into multiple subtasks and solve the problem of low efficiency of directly optimising the strategy of the overall goal in complex (DRL) tasks by learning hierarchical strategies and combining the strategies of multiple subtasks to form an effective overall strategy. This technique is commonly referred to as hierarchical DRL based on spatio-temporal abstraction and intrinsic motivation.

6 Conclusion

In this paper, we propose DRLB, a DRL-based load balancing mechanism for heterogeneous traffic in datacenter networks. DRLB distinguishes heterogeneous traffic and employs DRL technology for load balancing. Specifically, DRLB uses D4PG algorithm to learn the optimal load balancing policy, and dynamically adjusts LSM value via DRL agent to guide long flows to select the least congested forwarding paths. The most suitable link for forwarding addresses the issue that the traditional load balancing mechanism cannot adapt well to change traffic scenarios and employs WCMP mechanism to safeguard delay-sensitive short flows against the issue of a lengthy pre-learning period for DRL. Using the 99^{th} percentile FCT of short flows and the throughput of long flows as performance metrics, our experimental results demonstrate that DRLB significantly improves the overall performance and stability of the datacenter network when compared with the existing Freeway, Hermes, and TLB mechanisms. The algorithm can be further enhanced and validated in real-world scenarios, and DRLB can be refined in future research.

References

1. Li, Z., Bai, W., Chen, K.: Rate-aware flow scheduling for commodity data center networks. In Proc. IEEE INFOCOM, pp. 1-9 (2017)

2. Hu, J., He, Y., Wang, J., Luo, W., Huang. J.: RLB: Reordering-Robust Load Balancing in Lossless Datacenter Network. In Proc. ACM ICPP, (2023)
3. Ma, X., Hu, C., Chen, K., Zhang, C., Zhang, H., Zheng, K., Sun, X.: Error tolerant address configuration for data center networks with malfunctioning devices. In: Proceedings of IEEE INFOCOM, pp. 708-717 (2012)
4. Jing, Q., Wang, W., Zhang, J., Tian, H., Chen, K.: Quantifying the performance of federated transfer learning.arXiv preprint arXiv:1912.12795, (2019)
5. Hu, C., Chen, K., Chen, Y., Liu, B.: Evaluating potential routing diversity for internet failure recovery. In: Proceedings of IEEE INFOCOM, pp. 1-5 (2010)
6. 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)
7. 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)
8. Li, W., Chen, S., Li, K., Qi, H., Xu, R., Zhang, S.:Efficient Online Scheduling for Coflow-Aware Machine Learning Clusters. IEEE ransactions on Cloud Computing,10(4), 2564-2579 (2020)
9. Zeng, G., Bai, W., Chen, G., Chen, K., Han, D., Zhu, Y. : Combining ECN and RTT for datacenter transport. In: Proceedings of the First Asia-Pacific Workshop on Networking, pp. 36-42 (2017)
10. Hopps, C.: Analysis of an equal-cost multi-path algorithm. RFC 2992, (2000)
11. Wang, W., Sun, Y., Zheng, K., Kaafar, M.A., Li, D., Li, Z.: Freeway: Adaptively isolating the elephant and mice flows on different transmission paths. In: Proceedings of the IEEE International conference on network protocols,pp. 362-367 (2014)
12. Ghorbani, S., Yang, Z., Godfrey, P. B., Ganjali, Y., Firoozshahian, A.: DRILL: Micro load balancing for low-latency data center networks. In: Proceedings of ACM SIGCOMM, pp. 225–238 (2017)
13. Zhang, H., Zhang, J., Bai, W., Chen, K., Chowdhury, M.: Resilient datacenter load balancing in the wild. In: Proceedings of ACM SIGCOMM, pp. 253–266 (2017)
14. Hu, J., Huang, J., Lv, W., Li, W., Wang J., He, T.: TLB: Trafficaware Load Balancing with Adaptive Granularity in Data Center Networks. In: Proceedings of ACM ICPP, pp. 1-10 (2019)
15. 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)
16. Wang, J., Rao, S., Ying, L., Sharma, P.K., Hu, J.: Load Balancing for Heterogeneous Traffic in Datacenter Networks. Journal of Network and Computer Applications, Vol. 217, (2023)
17. 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)
18. 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), 1898-1912 (2023)
19. Mnih, V., Kavukcuoglu, K., Silver, D.: Playing atari with deep reinforcement learning. In: Proceedings of Workshops at the 26th Neural Information Processing Systems,pp. 201-220 (2013)
20. Zhou, J.L., Tewari, M., Zhu, M., Kabbani, A., Poutievski, L., Singh, A., Vahdat. A.: WCMP: Weighted cost multipathing for improved fairness in data centers. In: Proceedings of ACM SIGCOMM,pp. 1-14 (2014)
21. Barth-Maron, G., Hoffman, M.W., Budden, D.: Distributed distributional deterministic policy gradients. arXiv preprint arXiv:1804.08617, (2018)

22. 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)
23. 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)
24. Wei, W., Gu, H., Wang, K., Li, J., Zhang, X., Wang, N.: Multi-Dimensional Resource Allocation in Distributed Data Centers Using Deep Reinforcement Learning. In IEEE TNSM, pp. 1817-1829 (2023)
25. 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), pp. 3849-3865 (2023)
26. Wei, W., Gu, H., Deng, W., Xiao, Zhe., Ren, X.: ABL-TC: A Lightweight Design for Network Traffic Classification Empowered by Deep Learning, Neurocomputing, pp. 333–344 (2022)
27. Hu, C., Liu, B., Zhao, H.: DISCO: Memory Efficient and Accurate Flow Statistics for Network Measurement. In Proc. IEEE ICDCS, pp. 665-674 (2010)
28. Li, H., Zhang, Y., Zhang, Z.: Ursa: Hybrid Block Storage for Cloud-Scale Virtual Disks. In Proc. ACM EuroSys, pp. 1-17 (2019)
29. 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)
30. 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)
31. 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)
32. 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)
33. 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)
34. Wei, W., Fu, L., Gu, H., Zhang, Y., Zou, T., Wang, C., Wang, N. : GRL-PS: Graph embedding-based DRL approach for adaptive path selection. In: Proceedings of IEEE Transactions on Network and Service Management, pp. 1-1 (2023)
35. 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)
36. Gao, X., Kong, L., Li, W., Liang, W., Chen, Y., Chen, G. Traffic load balancing schemes for devolved controllers in mega data centers. In: Proceedings of IEEE Transactions on Parallel and Distributed Systems,pp. 572-585 (2017)
37. Hu, J., Huang, J., Lv, W., Zhou, Y., Wang, J., He, T. CAPS: Coding-based Adaptive Packet Spraying to Reduce Flow Completion Time in Data Center, In: Proceedings of IEEE INFOCOM, pp. 2294-2302 (2018)
38. Liu, J., Huang, J., Li, W., Wang, J.: AG: Adaptive switching granularity for load balancing with asymmetric topology in data center network. In: Proceedings of IEEE ICNP, pp. 1-11 (2019)
39. Schulman, J., Wolski, F., Dhariwal, P.: Proximal policy optimization algorithms. arXiv preprint, arXiv:1707.06347, (2017)
40. Silver, D., Huang, A., Maddison, C. J.: Mastering the game of Go with deep neural networks and tree search. Nature, pp. 484-489 (2016)