Enabling Traffic-differentiated Load Balancing for Datacenter Networks*

Jinbin Hu, Ying Liu, Shuying Rao, 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
{yingliu,shuyingrao}@stu.csust.edu.cn

Abstract. In modern datacenter networks (DCNs), load balancing mechanisms are widely deployed to enhance link utilization and alleviate congestion. Recently, a large number of load balancing algorithms have been proposed to spread traffic among the multiple parallel paths. The existing solutions make rerouting decisions for all flows once they experience congestion on a path. They are unable to distinguish between the flows that really need to be rerouted and the flows that potentially have negative effects due to rerouting, resulting in frequently ineffective rerouting and performance degradation. To address the above issues, we present a traffic-differentiated load balancing (TDLB) mechanism, which focuses on distinguishing flows that necessarily to be rerouted and employing corresponding measures to make optimize routing decisions. Specifically, TDLB detects path congestion based on queue length at the switches, and distinguishes the traffic that must be rerouted through the host pair information in the packet header, and selects an optimal path for rerouting. The remaining traffic remains on the original path and relies on congestion control protocols to slow down to alleviate congestion.The NS-2 simulation results show that TDLB effectively reduces tailing latency and average flow completion time (FCT) for short flows by up to 45% and 46%, respectively, compared to the state-of-the-art load balancing schemes.

Keywords: Datacenter networks · Load balancing · Traffic differentition · Multiple paths.

1 Introduction

With the stringent requirements of low latency and high throughput for various datacenter applications like high-performance computing, big data analyt-

* 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), the Changsha University of Science and Technology Graduate Innovation Project (CLSJCX23101).

ics, and Internet of Things (IOT), providing high-performance network services in datacenters is of paramount importance [1–5]. Modern datacenter networks (DCNs) widely adopt a multi-rooted tree network topology to establish multiple parallel paths among host pairs, providing high-bandwidth network transmission capabilities [6, 7]. In recent years, numerous outstanding network processing technologies have emerged to fully leverage the multipath characteristics of datacenter networks for achieving high-performance network transmission. Among these technologies, load balancing mechanisms play a critical role in datacenter networks by effectively distributing the workload across multiple paths, mitigating link failures, and maximizing the utilization of parallel paths, thereby improving link utilization and system performance [8–10].

In order to enhance performance in multiple paths transmission scenarios, various load balancing mechanisms have been proposed to improve the transmission performance of the network. ECMP (Equal Cost Multi-Path) [11], which is presently most widely used load balancing mechanism, balances traffic poorly under multiple paths. ECMP uses a hash algorithm to randomly select forwarding paths for traffic, balancing traffic to multiple parallel paths to reduce path load and network congestion, and achieve high chain utilization and low latency transmission. However, ECMP fails to take advantage of the congestion information and traffic characteristics of the network, and hash routing can easily cause hash conflicts between different data flows on the transmission path, resulting in load imbalance and exacerbating network congestion hotspots, wasting network bandwidth.

Therefore, many enhanced load balancing mechanisms have been proposed to address issues such as hash conflicts and congestion hotspots in ECMP. In contrast, other solutions adopt useful information on the network. CONGA [12] selects the optimal transmission path for data flows by collecting global congestion information in real-time. DRILL [13] uses the queue information of the local switch and selects the forwarding port based on the queue length. LetFlow [14] detects path congestion and randomly forwards traffic along different paths at fixed time intervals. Hermes [15] detects global congestion and performs rerouting based on path conditions and data flows status. They effectively resolve hash collision issues, to a certain extent reduce path congestion, and improve network performance through making full use of information on multiple paths.

However, despite the excellent performance of load balancing mechanisms in many aspects, previous solutions have had a few significant shortcomings. The existing load balancing mechanism only triggers rerouting when congestion is perceived, without distinguishing the traffic that truly needs to be rerouted [16–18]. As a result, it does not perform differentiation processing and directly reroutes all traffic on the congested path. These solutions have to some extent alleviated the problem of link workload, but coarse-grained rerouting solutions cannot solve path congestion. Specifically, a small amount of congested traffic has not effectively solved the original problem through load balancing and rerouting [19–21]. This is because when these congested flows are rerouted, they may cause congestion to spread and affect other transmission paths, unable to solve

congestion hotspots. In addition, it will increase the queuing latency of some flows, leading to an increase in flow completion time. Therefore, these congested flows can only be alleviated by reducing the transmission rate of the link through congestion control mechanisms [22, 23].

To address these issues, we aim to provide a more intelligent and efficient load balancing solution to further optimize network performance and improve user experience [24–26]. We propose a traffic-differentiated load balancing mechanism (TDLB) primarily based on the principle of distinguishing congested flows and prioritizing traffic. Firstly, we introduce a traffic recognition mechanism that accurately identifies the traffic that truly needs to be rerouted and performs special processing on it. Secondly, we have designed a priority scheduling algorithm that allocates network resources reasonably based on the importance and priority of traffic, ensuring that high priority traffic is processed with priority. Among these aspects, we focus on how to distinguish traffic that truly needs to be rerouted and how to minimize the disruption caused by rerouting operations [15, 16]. In addition, we conduct performance comparisons between symmetric and asymmetric network topologies to evaluate the performance of TDLB.

1) We conduct in-depth research and analysis on two main issues during path congestion, including the increased tailing latency caused by undifferentiated traffic that truly needs to be rerouted, and the longer flow completion time caused by reordering resulting from rerouting.
2) We determine the path status by evaluating the queue length of the switch and design a traffic recognition mechanism to differentiate between flows that contribute to congestion and flows that truly need to be rerouted.
3) We propose a fine-grained load balancing mechanism TDLB that effectively separates traffic on congested paths and implements different scheduling schemes for flows with different characteristics.
4) We implement TDLB using NS-2 simulation, which effectively improves transmission performance under congested conditions and significantly reduces tailing latency and average flow completion time (AFCT).

In Section 2, we introduce the main motivation of this paper. In Section 3, we provide a detailed overview of the design and its details. In Section 4, we conduct experimental simulations and analysis, including the analysis of relevant comparative results. In Section 5, we introduce some representative related work. In the final section, we summarize the work of this paper.

2 Motivation

However, current load balancing mechanisms (such as ECMP, CONGA, and Hermes) have a common problem in that they all reroute all flows on congested paths without distinguishing between those that truly need to be rerouted. In this scenario, there are two serious issues. Firstly, the traffic that truly needs to be rerouted may still choose the same path as the traffic that causes congestion, leading to an increase in tail latency. Secondly, rerouting increases the chance of

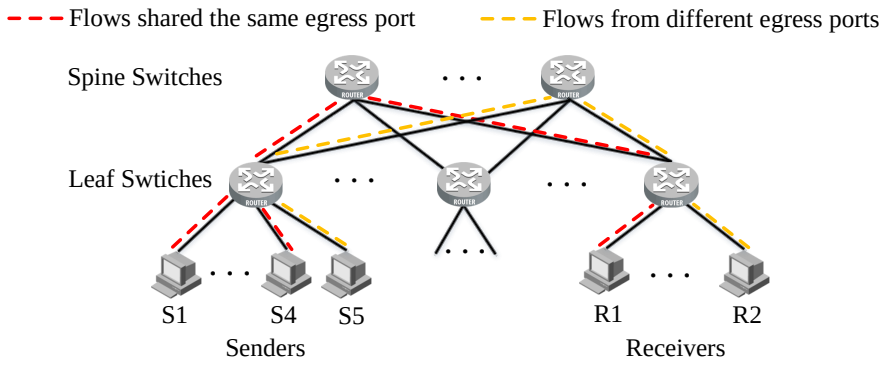


Fig. 1: Leaf-spine topology

disorder, which not only fails to reduce the flow completion time of congested flows but also leads to longer FCT [27, 28].

Study Case: In order to better illustrate the above issues in existing load balancing mechanisms, we will use a simple example to illustrate. As shown in Fig.1, this case adopts a common leaf-spine topology, where two leaf switches L1 and L2 are connected to two sending terminals (S1, S2) and two receiving terminals (R1, R2), respectively. The two leaf switches are fully connected to three spine switches, meaning there are three identical parallel paths between L1 and L2, which have the same link bandwidth and latency. S1 sends 4 flows ($f1 \sim f4$) to R1 at T1 moment, and S2 subsequently sends 1 flow $f5$ at T2 moment. We use five typical load balancing mechanisms to handle this network transmission scenario.

As depicted in Fig.2, whether it is ECMP at the flow level, LetFlow and CONGA also at the flow level, or DRILL and Hermes at the packet level, they may all be directed to the same path when handling flows ($f1 \sim f5$) on leaf switch L1. Moreover, a significant number of flows become congested in the queues of L2 switches, with 80% of the traffic being sent to the receiving terminal R1 and the remaining 20% to the receiving terminal R2. Consequently, this link experiences congestion, prompting the aforementioned load balancing mechanisms to redirect these flows and packets to the upper-level switch L1, employing different processing schemes to reroute this portion of traffic.

We analyz the impact of congestion on the completion time and tailing latency of short flows through NS-2 simulation testing [25]. We use a leaf-spine topology with three equal-cost paths between host pairs. The bottleneck bandwidth is 40Gbps, and the round-trip propagation latency is 10 μs. Each sender sends a DCTCP flow to the receiver through leaf and spine switches [23]. In this sample experiment, a mixture of a large number of flows from the same receiver and a small number of flows from other receivers are randomly generated under different network loads.

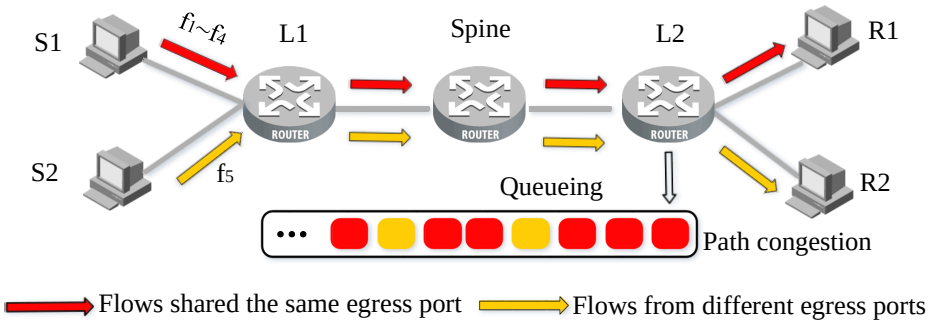
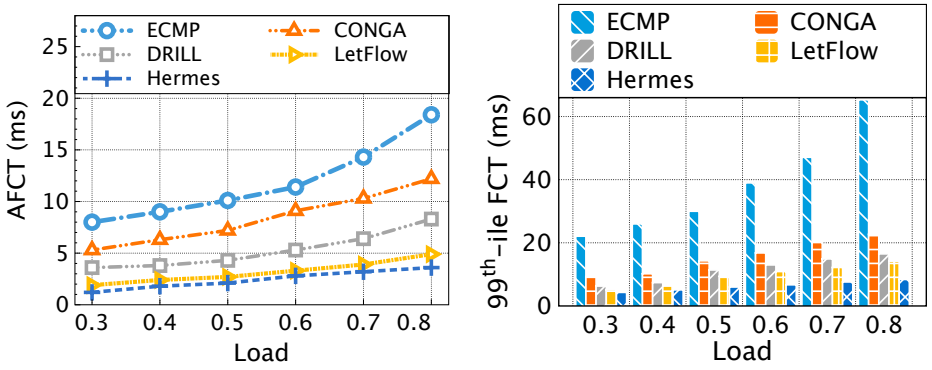


Fig. 2: Motivation example

A large number of flows ($f1 \sim f4$) with the same outbound port and a small number of flows $f5$ with different outbound ports are crowded on the same transmission path, where flows ($f1 \sim f4$) are sent from leaf switch L2 to receiver R1, while flow $f5$ is sent from leaf switch L2 to receiver R2. Obviously, this may lead to network performance bottlenecks and increases transmission latency. This is because a large amount of traffic shares the same transmission path, which may not be able to handle all traffic simultaneously. If not processed, a small amount of flows $f5$ sent to different receiving ports will be blocked in the output queue of leaf switch L2, seriously increasing the FCT of $f5$. Therefore, this situation requires a load balancing mechanism to schedule its flows.



(a) AFCT (b) 99^{th} percentile FCT

Fig. 3: Performance under different workloads

Large Tailing Latency. The existing load balancing mechanisms first perceive high load or congested paths and propagate all information on that path

back to the upstream switch. Then, according to the corresponding load balancing mechanism, this portion of traffic is rerouted to other parallel paths to reduce flow completion time and improve link utilization. As shown in Fig.3 (a), as the network load increases, AFCT continues to increase. Fig.3 (b) show the 99^{th} percentile FCT for different load balancing mechanisms as network load increases. We observed that the tailing latency is approximately twice the average flow completion time. It can be inferred that some flows have encountered severe congestion, despite load balancing and rerouting of congested paths, congestion still exists after switching paths. It is worth emphasizing that the tailing latency of ECMP is much higher than other load balancing mechanisms. This is because ECMP uses a hash method and does not have a congestion aware mechanism, only spreading the flow to different paths for forwarding. For paths that have already generated congestion, it is likely to exacerbate the congestion of the path, and generating hash collisions will also increase the possibility of link congestion. Overall, these congested flows are also rerouted and forwarded to other paths, which is likely to continue to cause congestion on other paths and ultimately increase the tailing latency of the flow.

Large Average Latency. The completion time of the flow actually increases. In Fig.2, the flows ($f1 \sim f4$) sent to the same receiving terminal experience congestion on both the receiving terminal and the last hop switch. In this case, these flows are processed through load balancing scheduling, and the traffic on the entire congested path is rerouted and forwarded to other parallel paths. Fig.3 (a) shows the average flow completion time under different load balancing mechanisms as the load increases. Despite rerouting the flow on congested paths, the overall average flow completion time has significantly increased. When the network load is 0.3, the AFCT of Hermes is 1.2 ms, which has reached a higher latency level compared to the link latency of 10 μs. ECMP mechanism based on hash routing has achieved an average flow completion time of 8 ms at a load of 0.3. This is because the ECMP does not utilize any information from the link, and the re-selected path is random, which may result in choosing a worse path. In addition, if the flow on a congested path is rerouted and redirected to another path, it may still face congestion, increasing the possibility of disorderly transmission [6, 14]. The flow that causes congestion cannot reduce the FCT by rerouting, on the contrary, disorderly arrival actually leads to an increase for the FCT.

In brief, we analyze the impact of existing load balancing mechanisms on network performance under congested conditions and come to the following conclusions: (i) Flows that truly need rerouting can still potentially choose the same path as the congesting traffic, resulting in increasing tail latency. (ii) Congestion-induced flows, after rerouting, increase the chances of packet reordering and are unable to reduce flow completion time, leading to longer completion times. These conclusions drive us to propose a congestion-aware load balancing mechanism that distinguishes between flows that genuinely require rerouting and congested flows, aiming to achieve low-latency, high-performance transmission.

3 TDLB Design

3.1 Design Overview

In this section, we provide an overview of the design of TDLB. The key to TDLB is to distinguish the traffic that needs to be rerouted, flexibly select the ideal route for this portion of the traffic, and effectively adjust the traffic on congested links. Specifically, on the one hand, identifying flows that truly need to be rerouted and rerouting them to other better parallel paths may reduce serious queuing and tailing latency. On the other hand, to prevent the traffic that causes congestion from being rerouted to other paths, causing congestion spread and affecting the normal transmission of other paths, and to prevent rerouting from increasing the chance of disorder and increasing the FCT. Fig.4 demonstrates the framework of TDLB, which comprises of three components.

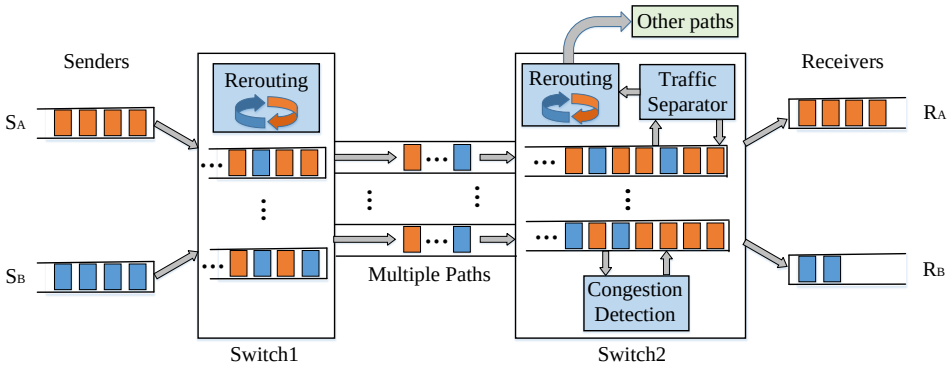


Fig. 4: The architecture of TDLB

1) **Congestion Detection:** TDLB performs congestion detection on the switch that is the last hop from the receiving terminal. Firstly, determine whether congestion has occurred on the path according to the queue length of the switch. Subsequently, the switch sends link congestion notifications to prevent the path congestion from becoming more serious [17, 20].

Specifically, switches usually establish a queue for each port to store incoming packets. Through monitoring the queue length of the switch (i.e., the total amount of packets in the queue), it is possible to inadvertently figure out whether congestion has occurred on this path. When the queue length exceeds the threshold set in advance, congestion can be regarded as having occurred.

2) **Traffic Separation:** The traffic separator is employed to separate congested traffic and other harmless traffic on congested paths, and to forward innocent traffic to other parallel paths through the rerouting mechanism on the local switch.

When the switch detects congestion, it indicates that some traffic has affected the normal transmission of the path, which is the flow causing congestion. Only

when congestion occurs in the link, TDLB will employ the traffic separator to distinguish between the congested traffic and other innocent traffic, and deal with them in a fine-grained method to alleviate path congestion.

3) **Truly Necessary Rerouting:** For innocent traffic on congested paths, selects a better transmission path based on the current network state and the queue length of the switch forwarding port.

To further alleviate network congestion, TDLB transforms its innocent flows from congested paths to other available non-congested paths. This can reduce the workload on network congestion, improve overall network performance and service quality.

3.2 Design Rationale

In the current load balancing mechanism, there is no distinguishing flows on the congested path that truly needs to be rerouted. They are forwarded to other parallel paths in the same method, resulting in increasing congestion and tailing latency. Specifically, when a path becomes congested, not all traffic is negative. Only a portion of the traffic is actually causing congestion, while the remaining traffic is innocent normal traffic. If no processing is carried out, the traffic causing congestion will continue to block the queue exit, and innocent traffic waiting in the queue will also become a criminal flow causing congestion. This will undoubtedly increase the FCT of traffic on the path, as well as the tailing latency. The load balancing mechanism for this case will forward every flow on the congested path to the higher-level switch, which will subsequently reroute depending on a specific routing algorithm. The existing load balancing mechanisms reroutes all flows without distinguishing between congested and innocent flows, which can cause congestion spread in other paths and increase the possibility of packet disorder, increasing the FCT of congested traffic.

We address these issues from different perspectives focusing on the topic of whether it is possible to explore a novel load balancing mechanism with the following design goals: (i) Detecting path congestion in a more effective and quicker manner; (ii) Designing an efficient flow separator to distinguish traffic that truly need to be rerouted; (iii) Making sure rerouting to better parallel paths to achieve efficient network load balancing.

3.3 Design Detail

In this section, we provide a detailed explanation of the key designs in each part of TDLB.

Congestion Detection: The switch maintains a queue for storing packets, and whenever a packet arrives at the switch and cannot be immediately forwarded, it will be added to the queue [30, 31]. When the path becomes congested, the switch ports become blocked, and the queue length gradually increases. At the same time, the path delay also increases. If relying solely on queue length to perceive congestion, it may lead to misjudgment or omission. Therefore, the

exchange opportunity regularly checks the length of the queue and sets a threshold of Φ for the queue length in advance. When the length of the queue exceeds this threshold, we believe it may cause congestion. To ensure the accuracy of congestion detection, we also need to measure whether the path delay changes when the queue length exceeds the threshold. When the threshold is exceeded and the path delay increases, we determine that congestion has occurred. We set Φ to the half of queue length. It is worth noting that this article adopts DCTCP [23] as the congestion control algorithm.

Traffic Separation: The existing load balancing mechanism is unable to differentiate traffic in congested cases, and coarse-grained routing leads to an increase in FCT and tail latency [28, 32, 33]. Inspired by the distinction between long and short flows to reduce FCT of short flows, we differentiate between congested and innocent flows and perform fine-grained processing based on the characteristics of the traffic. When congestion is detected, distinguish the flow that truly needs to be rerouted to achieve the separation of the congested flow from the innocent traffic.

Specifically, we identify the traffic causing congestion by analyzing traffic characteristics and utilizing traffic statistics data. Firstly, it is necessary to establish classification rules for the traffic in the switch and classify it based on the characteristics of the traffic. Generally speaking, traffic can be distinguished by five tuples (source IP address, destination IP address, source port number, destination port number, protocol type). In TDLB, we have developed a hash array for distinguishing congested traffic to record the hash value and quantity of traffic quintuples. At the same time, a congestion traffic flag has been added to the header of the data packet.

Algorithm 1: Distinguishing congested flows algorithm

Input: A packet belongings to flow f
Output: Congestion flow tag T
if *cur_ queue_ length >queue_ threshold and cur_ path_ delay >default_ delay* **then**
    Record current congestion flows into the **array**;
    Mark congestion flow tag $T \rightarrow$ True ;
end
for *every packet* **do**
    Assume the corresponding flow is f;
    if *hash(f) exists in array* **then**
        Mark congestion flow tag $T \rightarrow$ True ;
    else
        Mark congestion flow tag $T \rightarrow$ False
    end
    return T
end

The mechanism for distinguishing congested flows is shown in Algorithm 1. When the switch detects congestion, the hash array will treat all recorded traffic as causing the congestion. After other subsequent traffic through hash operations, check if the hash value exists in the array. If it does, mark the data packet with congestion, and increase the count corresponding to the hash value by one; Otherwise, it is considered innocent traffic and truly needs to be rerouted.

Truly Necessary Rerouting: The rerouting rules play a crucial role in the performance of load balancing mechanisms [34, 35]. It ensures high reliability transmission of the network through reselecting other parallel paths in the event of link congestion or network topology changes. The switch maintains a routing table for storing routing information in the network. When a link failure or network topology change occurs, the switching opportunity updates the routing table accordingly to reflect the new network state.

The rerouting operation is used by local switches experiencing congestion to handle a small amount of innocent traffic on congested paths. In order to better handle the differences between congested and non congested flows, a congestion separation based rerouting algorithm is proposed. TDLB uses a traffic separation mechanism to create congestion markers for congested and innocent flows, which can clearly distinguish congested flows on the path. Congestion flows will be assigned lower priority and kept on the path waiting for congestion to end, in order to limit their transmission rate and reduce the impact on network congestion; Innocent flows will be assigned higher priority to obtain better network resources. TDLB reroutes these innocent flows without congestion markers and forwards them to good paths to ensure better quality of service.

The rerouting mechanism of TDLB is deployed on the incoming port of the switch, determining the forwarding path based on the current queue length and link delay of the switch, in order to route traffic that truly necessary rerouting to other better path. Specifically, when data packets enter the switch, the detected innocent traffic will trigger the rerouting mechanism to be forwarded to other better parallel paths. For the selection of parallel paths, it is determined based on the queue length of the switch's output port, scanning to find the output port with the smallest queue length, which is the current optimal path. It is worth noting that using only queue length for rerouting cannot effectively determine the optimal path. Therefore, when the queue length of different outgoing ports in the switch is the same, it is also necessary to select a path with smaller latency through link latency. By comparing the queue length and link delay of multiple paths, a better forwarding path can be obtained and congestion on that path can be effectively alleviated and eliminated.

4 Evaluation

We conduct experiments with two classic workloads, Web Search and Data Mining, to evaluate the performance of TDLB in large-scale scenarios [6, 36]. In the above workloads, the distribution of flow size is heavy-tailed, and the specific flow size distribution is shown in Table 1. In general, flows smaller than 100KB

are regarded as short flows, while others are considered long flows. Due to the small size of short flows, special attention is paid to the AFCT, while long flows are more concerned with higher throughput. In terms of traffic size distribution, under Web Search workload, approximately 20% of traffic is greater than 1MB, while in Data Mining workload, less than 5% of traffic is greater than 35MB. Web Search and Data Mining have 62% and 83% of short flows smaller than 100KB, respectively, resulting in a heavy-tailed distribution.

Table 1: Flow size distribution of two workloads

	Web Search	**Data Mining**
0-100KB	62%	83%
100KB-1MB	18%	8%
>1MB	20%	9%
Average flow size	1.6MB	7.4MB

The flow size distribution of Web Search and Data Mining usually presents a long tail distribution. For Web Search, this means that a few popular search terms will account for the majority of traffic, while most search terms only contribute a small proportion of traffic. Although the proportion of long wake flow is relatively low, they still have certain importance. Long wakes may play a crucial role in specific scenarios or problems. Therefore, it is necessary to pay special attention to the transmission quality of long wakes in the network to avoid the negative impact of severe tailing on network performance as much as possible.

We compare TDLB with five state-of-the-art load balancing mechanisms, including ECMP, CONGA, DRILL, LetFlow, and Hermes. ECMP uses static hashing to allocate each flow to an equivalent multipath. CONGA selects the optimal transmission path for data flows through real-time feedback of global congestion information. DRILL selects the port for packet forwarding based on the length of the local queue. Letflow classifies flowlets using fixed time intervals and randomly selects forwarding ports for each flowlet. Hermes reroutes based on path conditions and data flows status.

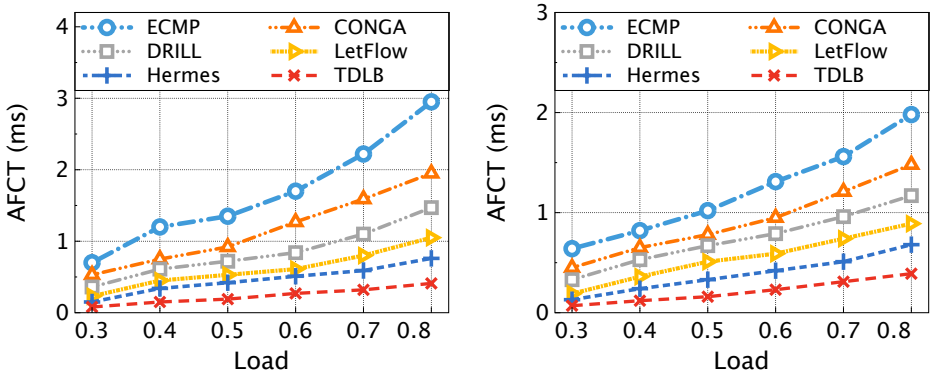
We still use the same leaf-spine topology as the motivation as the experimental network topology, which consists of 8 leaf switches and 9 spine switches. Each leaf switch connects 32 terminals, so the entire network has 256 independent terminals connected through a 40Gbps bandwidth link, with a round-trip propagation latency of 10us. The traffic in the network is generated randomly through the Poisson distribution between random host pairs. We adjust the workload from 0.3 to 0.8 to thoroughly evaluate the performance of TDLB.

In the following two experiments, we use symmetric and asymmetric network topologies for performance evaluation. A symmetric topology structure means that the connection modes between nodes in the network are the same, that is,

the bandwidth and latency on each link are the same. Therefore, a symmetric topology structure is easier to achieve load balancing in the network. On the contrary, the connections and characteristics between network nodes in asymmetric topology structures are different, and the bandwidth and latency on different paths may vary. Therefore, if you want to achieve high-quality transmission performance in an asymmetric topology network, it requires a more flexible and scalable load balancing mechanism.

4.1 Performce Under Symmetric Topology

In this section, we test the performance of TDLB on the AFCT, 95^{th} percentile FCT and 99^{th} percentile FCT of short flows in network transmission. Compared with five load balancing mechanisms under Web Search and Data Mining workloads.



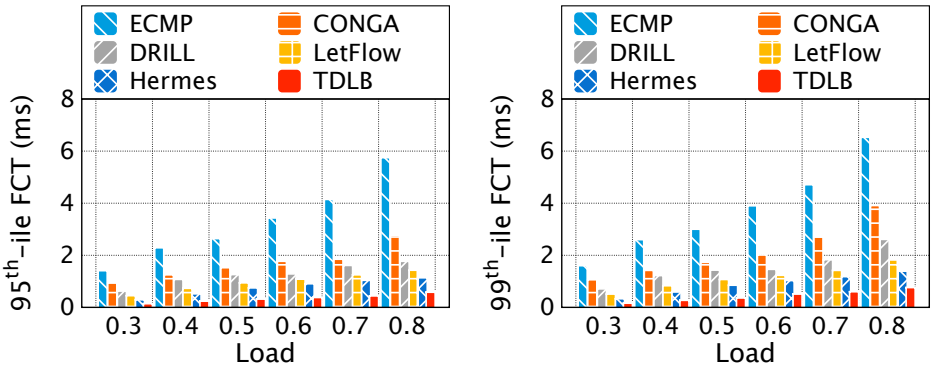
(a) Web Search workload (b) Data Mining workload

Fig. 5: Comparison of AFCT in different workloads under symmetric topology

Fig.5 (a) and Fig.5 (b) show the average FCT of short flows under Web Search and Data Mining workloads, respectively. Compared with the other five load balancing mechanisms, TDLB significantly reduces AFCT, especially in high workload situations. Specifically, under a 0.8 workload, in the Web Search scenario, TDLB decreases by $\sim$ 86%, $\sim$ 79%, $\sim$ 72%, $\sim$ 61%, and $\sim$ 46%, compared to ECMP, CONGA, DRILL, LetFlow, and Hermes, respectively. In the Data Mining scenario, compared to ECMP, CONGA, DRILL, LetFlow, and Hermes, TDLB decreases by $\sim$ 80%, $\sim$ 73%, $\sim$ 67%, $\sim$ 56%, and $\sim$ 44%, respectively.

These test results indicate that TDLB performs well in low latency transmission. Within the same switch, as workload increases, more mixed flows queue up in the output queue, leading to path congestion. Consequently, this results in the blocking of additional traffic that has just arrived at the path, causing

it to experience significant queuing latency and packet reordering. Specifically, ECMP operates at the flow level and experiences long queuing latency in cases of path congestion. Similarly, LetFlow operates solely at the flow level but utilizes random routing. The feedback latency of CONGA in obtaining congestion information increases, leading to the creation of long queues. While DRILL can handle sudden congestion in a timely manner, it struggles to accurately perceive global congestion information and avoid the retransmission cost caused by disorder. The rerouting mechanism triggered by Hermes is overly conservative, often leading to low link utilization.



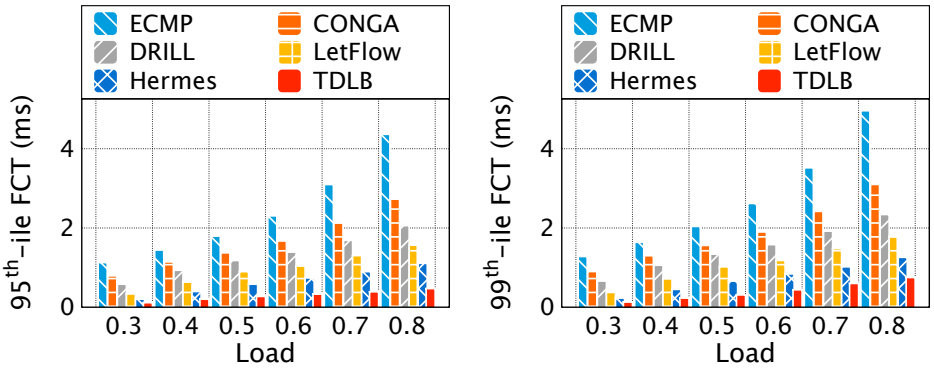
(a) 95^{th} percentile FCT (b) 99^{th} percentile FCT

Fig. 6: Comparison of tailing latency in Web Search workload under symmetric topology

TDLB, proposed in this paper, first has the ability to perceive congestion and detect sudden congestion in a timely manner. Secondly, based on the characteristics of the flow, it identifies the flow that truly needs redirection and forwards it to a better path. This effectively solves the original path congestion problem and prevents congestion diffusion. The congested flows continue to queue and wait on the original path to avoid rerouting causing disorder and increasing the completion time of the flow. Non-congested innocent flows truly need to be rerouted to find better transmission paths, accelerate the arrival time of the flow, and reduce the completion time of the flow. Therefore, TDLB can achieve a lower average flow completion time.

In addition, we find that the short flows in the Web Search workload is larger than the FCT in Data Mining. The reason is that in Web Search, there are more flows ranging in size from 100KB to 1MB, accounting for 18% of the total traffic, resulting in longer queue lengths for switch queues and increasing opportunities for packet disorder. In Data Mining, flows smaller than 100KB account for about 83%, and a small amount of long flows will reduce the latency

of short flows queuing. Under the Web Search workload, the proportion of long flows is heavier, and during transmission, long flows occupy the transmission path, while short flows experience longer queuing latency, resulting in a larger overall AFCT.



(a) 95^{th} percentile FCT (b) 99^{th} percentile FCT

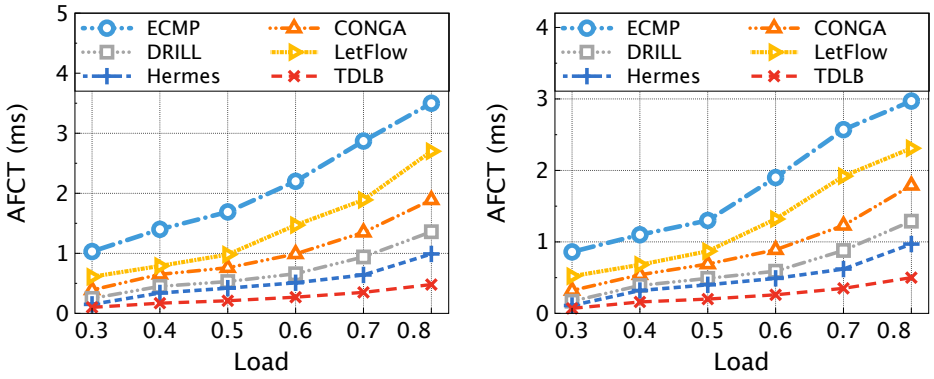
Fig. 7: Comparison of tailing latency in Data Mining workload under symmetric topology

Fig.6 (a), Fig.(b), Fig.7 (a), and Fig.(b) show the 95^{th} percentile FCT and 99^{th} percentile FCT of short flows under Web Search and Data Mining workloads, respectively. Obviously, compared to the other five solutions, TDLB significantly improves the tail FCT, especially in high workload situations. Specifically, under a 0.8 workload, in the Web Search scenario, TDLB decreases by $\sim$ 88%, $\sim$ 80%, $\sim$ 70%, $\sim$ 58%, and $\sim$ 45% compared to ECMP, CONGA, DRILL, LetFlow, and Hermes, respectively. In the Data Mining scenario, compared to ECMP, CONGA, DRILL, LetFlow, and Hermes, TDLB decreases by $\sim$ 84%, $\sim$ 76%, $\sim$ 68%, $\sim$ 58%, and $\sim$ 40% respectively.

Severe tailing refers to a situation where the latency of a small portion of the flow is significantly higher than the average latency. This phenomenon may have a negative impact on network performance. ECMP randomly select forwarding paths, without considering the congestion situation of other parallel paths, and transmit at the flow level. Therefore, ECMP has the greatest tailing effect. When the load balancing mechanism is unable to evenly distribute the load onto available servers, it may lead to some servers being overloaded, leading to serious latency. TDLB can truly redirect the flow that needs to be redirected and reroute it to a more optimal parallel path. This enables those innocent flows to complete transmission as early as possible, reducing overall flows latency.

Furthermore, we are surprised to find that under the same workload, the 99^{th} percentile FCT is approximately twice that of AFCT. We conduct a detailed

theoretical analysis of the experimental results. Firstly, the 99^{th} percentile FCT represents the portion of the flow with the highest tailing latency in transmission. Under normal case, the traffic is transmitted at the maximum transmission rate of the transmission path, and its FCT is less than or equal to AFCT. However, when the network load increases, it is inevitable that congestion will occur, resulting in some queuing latency and increasing the FCT. Meanwhile, prolonged waiting or excessive congestion can lead to packet loss and disorder. In general congestion situations, data packets only need to be retransmitted once to restore normal transmission of the flow. Therefore, the tailing latency includes the transmission latency before and after retransmission, which is approximately twice that of AFCT.



(a) Web Search workload (b) Data Mining workload

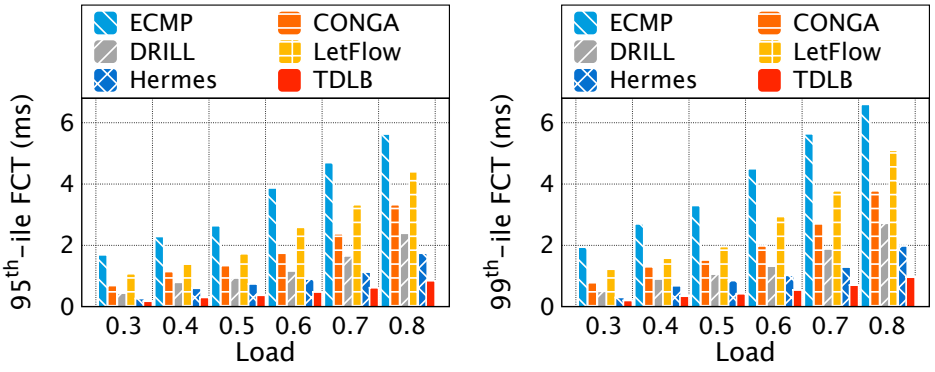
Fig. 8: Comparison of AFCT in different workloads under asymmetric topology

4.2 Performance Under Asymmetric Topology

In this section, we test the performance of TDLB under asymmetric network topology. Firstly, we introduce the characteristics of asymmetric network topology, and then conducted a detailed analysis of the experimental results.

Fig.8 (a) and Fig.8 (b) show AFCT under Web Search and Data Mining workloads, respectively. From the experimental results, ECMP is still the worst, followed by LetFlow, while TDLB still performs best and can achieve excellent performance in asymmetric topology scenarios.

In asymmetric topology, due to the differences between nodes, some nodes need to carry more traffic. In the case of unequal link bandwidth, when the network data volume increases, the node with unequal bandwidth between the uplink and downlink links will cause serious congestion at that node due to the significant difference in acceptance rate and transmission rate [37, 38]. Due

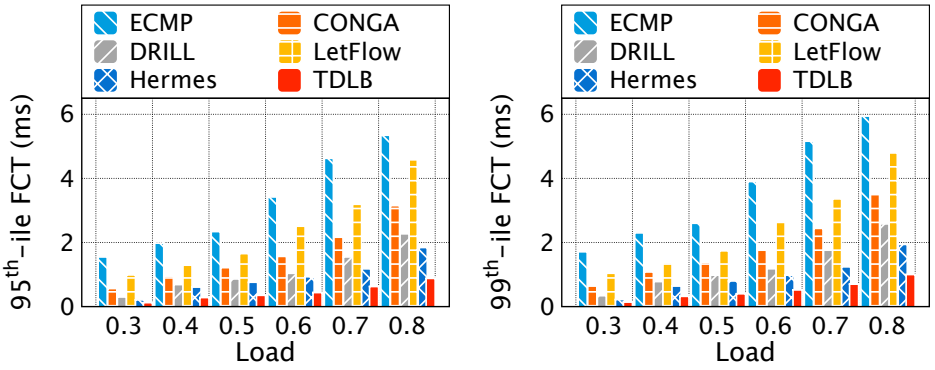


(a) 95^{th} percentile FCT (b) 99^{th} percentile FCT

Fig. 9: Comparison of tailing latency in Web Search workload under asymmetric topology

to ECMP and LetFlow randomly selecting paths through hash scattering and fixed time intervals, the uncertainty caused by random path selection may be forwarded to paths with lower link bandwidth, resulting in excessive congestion and FCT elongation. CONGA, DRILL, and Hermes utilize the congestion information of the path to alleviate the sudden congestion on the current path, but the feedback latency of global congestion perception was too large. The switch is unable to sense the congestion status of other paths in a timely manner, and cannot reroute traffic on congested paths to appropriate paths, which may increase the load on instantaneous congested paths and ultimately lead to longer FCT. At the same time, the unequal bandwidth and latency of asymmetric topology links can lead to more severe packet disorder due to partial rerouting, which affects the original transmission latency.

In addition, they do not distinguish the traffic that needs to be rerouted before rerouting, which can cause congestion when the traffic that causes congestion flows through rerouting and is forwarded to other paths to continue to generate congestion. In congested case, TDLB distinguishes innocent flows that truly need to be rerouted and forwards them to better paths based on link congestion information, thereby reducing the FCT of innocent flows. The congested flow continues to queue on the original path, replacing rerouting with speed reduction to avoid disorder and significantly reduce the FCT of the congested flow. Fig.9 (a), Fig.9 (b), Fig.10 (a) , and Fig.10 (b) respectively show the 95^{th} percentile FCT and 99^{th} percentile FCT under the Web Search and Data Mining workloads with asymmetric topology. When the workload in the Web Search scenario is 0.8, the 99^{th} percentile FCT of TDLB is 0.96 ms, which increases TDLB decreases by $\sim$ 85%, $\sim$ 75%, $\sim$ 65%, $\sim$ 81%, and $\sim$ 52% compared to ECMP, CONGA, DRILL, LetFlow, and Hermes, respectively. Overall, TDLB can also exhibit excellent performance in asymmetric topological structures.



(a) 95^{th} percentile FCT (b) 99^{th} percentile FCT

Fig. 10: Comparison of tailing latency in Data Mining workload under asymmetric topology

5 Related Work

There are many literature on datacenter load balancing [6, 11–16, 35, 36, 39, 40]. Among them, we only discuss and summarize some representative work related to this paper.

ECMP [11] achieves load balancing by distributing network traffic across multiple paths with equal costs. When a packet reaches the load balancer or router, the ECMP mechanism utilizes predefined load balancing algorithms, such as hash or round-robin algorithms, to select an optimal path from the available options for transmitting the packet. While ECMP effectively achieves load balancing, it is not without limitations. It may suffer from line-rate congestion and tail latency issues, particularly when dealing with short and long flows. Additionally, the presence of hash collisions can lead to suboptimal utilization of available bandwidth.In comparison, RPS [16] offers a straightforward packet-level load balancing mechanism by randomly distributing packets among all available transmission paths on the switch. This technique enhances link utilization and ensures better distribution of traffic. However, it also has inherent drawbacks. Packet reordering issues may arise, potentially impacting the order of packet delivery. Moreover, RPS lacks the ability to proactively detect congestion, which can result in packet loss and subsequent retransmissions.

CONGA [12] achieves global congestion-aware load balancing by leveraging end-to-end path feedback information. It effectively detects congestion and failure in paths, making it suitable for asymmetric networks. However, the implementation of CONGA presents challenges due to the substantial storage requirements for path information and the reliance on custom switches, limiting its scalability. Additionally, the accuracy of feedback obtained from remote switches may be compromised, leading to potential discrepancies in real-time path con-

ditions. HULA [40] tackles the scalability issues of CONGA by employing programmable switches for congestion-aware load balancing. Nevertheless, HULA's strategy of selecting solely the optimal next-hop path may give rise to herd effects, resulting in congestion along the chosen path. Moreover, the update rate of the best path depends on the probing frequency, which, if excessively used, can degrade network efficiency.

To mitigate the overhead associated with switch-based congestion detection, LetFlow [14] autonomously senses path congestion based on intrinsic packet characteristics. Compared to global congestion-aware schemes like CONGA, LetFlow eliminates the need for comprehensive global congestion information, enhancing its scalability. However, the stochastic nature of LetFlow scheduling prevents it from achieving optimal load balancing performance. To efficiently alleviate congestion caused by bursty traffic, DRILL [13] introduces a load balancing strategy specifically designed for micro-bursts. By leveraging local queue lengths in switches, DRILL swiftly performs packet-level forwarding decisions, enabling microsecond-level load balancing and effectively resolving congestion issues arising from micro-bursts. Nevertheless, DRILL exhibits limitations when confronted with high-speed bursts. Hermes [15] utilizes ECN, latency, and probing packets to detect path congestion and failure states, estimating the benefits of switching paths to determine if rerouting should be executed. This approach mitigates the negative consequences associated with blind path switching, such as packet reordering, while also circumventing the mismatch between link states and congestion windows. Although Hermes is applicable to asymmetric networks, its conservative rerouting decision-making process can lead to suboptimal link utilization.

6 Conclusion

We present a traffic-differentiated load balancing mechanism called TDLB to solve the problem of large tailing latency caused by coarse-grained rerouting in existing load balancing mechanisms. TDLB aims at distinguishing traffic that truly needs to be rerouted and taking different measures to optimize the routing and transmission of traffic on congested paths. Firstly, the switch senses whether the current path is congested based on queue length and path delay. Secondly, when the path becomes congested, TDLB distinguishes between the flows causing congestion and the ones that truly need to be rerouted. Finally, the flow that truly needs to be rerouted will select a better transmission path based on the congestion information and queue length of other parallel paths. The experimental results of NS-2 simulation show that compared with existing mechanisms, TDLB can decrease the tailing latency and AFCT by to 45% and 46%, respectively, supplying more effectively network performance.

References

1. 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), 2564-2579 (2020)
2. Wang, J., Liu, Y., Rao, S., Zhou, X., Hu, J.: A Novel Self-adaptive Multi-strategy Artificial Bee Colony Algorithm for Coverage Optimization in Wireless Sensor Networks. Ad Hoc Networks, 150, pp. 103284 (2023)
3. Li, H., Zhang, Y., Li, D., et al.: Ursa: Hybrid block storage for cloud-scale virtual disks. In: Proceedings of the Fourteenth EuroSys Conference, pp. 1-17 (2019)
4. Wang, J., Liu, Y., Rao, S., et al.: Enhancing security by using GIFT and ECC encryption method in multi-tenant datacenters. Computers, Materials & Continua, **75**(2), 3849-3865 (2023)
5. Wang, Y., Wang ,W., Liu, D., et al.: Enabling edge-cloud video analytics for robotics applications. IEEE Transactions on Cloud Computing, **11**(2), 1500-1513 (2023)
6. Wang J., Rao S., Liu Y., et al.: Load balancing for heterogeneous traffic in datacenter networks. Journal of Network and Computer Applications, 217 (2023)
7. Hu, J., Zeng, C., Wang, Z., et al.: Enabling Load Balancing for Lossless Datacenters. In: Proceedings of IEEE ICNP (2023)
8. 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)
9. 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)
10. Bai, W., Chen, K., Hu, S., Tan, K., Xiong, Y.: Congestion control for high-speed extremely shallow-buffered datacenter networks. In: Proceedings of ACM APNet, pp. 29-35 (2017)
11. Hopps, C.E.: Analysis of an equal-cost multi-path algorithm (2000)
12. Alizadeh, M. et al.: CONGA: Distributed congestion-aware load balancing for datacenters. In Proc. ACM Conf. SIGCOMM, pp. 503–514 (2014)
13. Ghorbani, S., Yang, Z., Godfrey, P. B., Ganjali, Y., and Firoozshahian, A.: DRILL: Micro load balancing for low-latency data center networks. In Proc. ACM SIGCOMM, pp. 225-238 (2017)
14. Vanini, E., Pan, R., Alizadeh, M., Taheri, P., and Edsall, T.: Let it flow: Resilient asymmetric load balancing with flowlet switching. In Proc. USENIX NSDI, pp. 407-420 (2017)
15. Zhang, H., Zhang, J., Bai, W., Chen, K., and Chowdhury, M.: Resilient datacenter load balancing in the wild. In Proc. ACM SIGCOMM, pp. 253-266 (2017)
16. Dixit, A., Prakash, P., Hu, Y. C., and Kompella, R. R.: On the impact of packet spraying in data center networks. In Proc. IEEE INFOCOM, pp. 2130-2138 (2013)
17. 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)
18. 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)
19. Kabbani, A., Vamanan, B., Hasan, J., Duchene, F.: FlowBender: Flow-level adaptive routing for improved latency and throughput in datacenter networks. In Proc CoNEXT, pp. 149-160 (2014)
20. Wang, J., Yuan, D., Luo, W., et al.: Congestion control using in-network telemetry for lossless datacenters. Computers, Materials & Continua, **75**(1), 1195-1212 (2023)
21. Wen, K., Qian, Z., Zhang, S., Lu, S.: OmniFlow: Coupling load balancing with flow control in datacenter networks. In Proc. ICDCS, pp. 725-726 (2016)

22. Shafiee, M., Ghaderi, J.: A simple congestion-aware algorithm for load balancing in datacenter networks. In Proc. INFOCOM, pp. 1-9 (2016)
23. Alizadeh, M., Greenberg, A. et al: Data center TCP (DCTCP). In Proc. ACM SIGCOMM, pp. 63-74 (2010)
24. Munir, A., Qazi, I. A., Uzmi, Z. A., Mushtaq, A., Ismail, S. N., Iqbal, M. S., and Khan, B.: Minimizing flow completion times in data centers. In Proc. INFOCOM, pp. 2157-2165 (2013)
25. Li, Z., Bai, W., Chen, K., et al.: Rate-aware flow scheduling for commodity data center networks. In: Proceedings of IEEE INFOCOM, pp. 1-9 (2017)
26. David, Z., Tathagata, D., Prashanth, M., Dhruba, B., and Randy, K.: DeTail: Reducing the flow completion time tail in datacenter networks. In Proceedings of the ACM SIGCOMM, pp. 139–150 (2012)
27. Benson, T., Akella, A., and Maltz, D.: Network traffific characteristics of data centers in the wild. In Proc. ACM IMC, pp. 267-280 (2010)
28. Hu, C., Liu, B., Zhao, H., et al.: Discount counting for fast flow statistics on flow size and flow volume. IEEE/ACM Transactions on Networking, **22**(3), 970-981 (2013)
29. The NS-2 network simulator. http://www.isi.edu/nsnam/ns.
30. Bai, W., Hu, S., Chen, K., Tan, K., Xiong, Y.: One more config is enough: Saving (DC) TCP for high-speed extremely shallow-buffered datacenters. IEEE/ACM Transactions on Networking, **29**(2), 489-502 (2020)
31. Liu, Z., Chen, K., Wu, H., Hu, S., Hut, Y. C., Wang, Y., Zhang, G.: Enabling work-conserving bandwidth guarantees for multi-tenant datacenters via dynamic tenant-queue binding. In IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pp. 1-9 (2018)
32. Hu, C., Liu, B., Zhao, H., Chen, K., et al.: Disco: Memory efficient and accurate flow statistics for network measurement. In: Proceedings of IEEE ICDCS, pp. 665-674 (2010)
33. Wei, W., Gu, H., Deng, W., Xiao, Z., Ren, X.: ABL-TC: A Lightweight Design for Network Traffic Classification Empowered by Deep Learning, Neurocomputing, 489, 333–344 (2022)
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, IEEE Transactions on Network and Service Management (2023)
35. Hu, J., He, Y., Wang, J., et al.: RLB: Reordering-Robust Load Balancing in Lossless Datacenter Network. In: Proceedings of ACM ICPP (2023)
36. 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) Wang J, Rao S, Liu Y, et al.: Load balancing for heterogeneous traffic in datacenter networks. Journal of Network and Computer Applications, 217 (2023)
37. Zhao, Y., Huang, Y., Chen, K., Yu, M., et al.: Joint VM placement and topology optimization for traffic scalability in dynamic datacenter networks. Computer Networks, 80, 109-123 (2015)
38. Zheng, J., Du, Z., Zha, Z., et al.: Learning to Configure Converters in Hybrid Switching Data Center Networks. IEEE/ACM Transactions on Networking, 1-15 (2023)
39. 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)
40. Katta, N., Hira, M., Kim C, Sivaraman A, Rexford J.: HULA: Scalable load balancing using programmable data planes. In Proceedings of the Symposium on SDN Research, pp. 1-12 (2016)