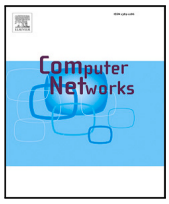
Computer Networks 255 (2024) 110867

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Towards fine-grained load balancing with dynamical flowlet timeout in datacenter networks☆

Jinbin Hu [a,b], Ruiqian Li [a], Ying Liu [a], Jin Wang [c,*]

[a] School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410004, China
[b] Key Lab of Broadband Wireless Communication and Sensor Network Technology (Nanjing University of Posts and Telecommunications), Ministry of Education, Nanjing 210003, China
[c] Sanya Institute of Hunan University of Science and Technology, Sanya 572024, China

ARTICLE INFO

Keywords:
Datacenter networks
Load balancing
Traffic differentiation

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. Fine-grained rerouting will also cause severe packet reordering, especially in asymmetric topology scenarios. To address the above issues, we present a fine-grained traffic-differentiated load balancing (TDLB) mechanism, which aims to distinguish flows that are necessarily to be rerouted and reroute traffic in fine-grained without packet reodering. Specifically, TDLB distinguishes the traffic that must be rerouted through the host pair information in the packet header, and selects an optimal path for rerouting. To prevent severe packet reodering caused by excessive path delay differences, TDLB dynamically adjusts the flowlet timeout to segment the traffic and select the optimal path for rerouting. The NS-2 simulation results show that TDLB effectively reduces tail latency and average flow completion time (FCT) for short flows by up to 49% and 46%, respectively, compared to the state-of-the-art load balancing schemes.

1. Introduction

With the stringent requirements of low latency and high throughput for various datacenter applications like high-performance computing, big data, 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 path and maximizing the utilization of parallel paths, thereby improving the system performance [8–10].

In order to enhance performance in multiple path 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 the most widely used load balancing mechanism, balances traffic poorly under multiple path. ECMP uses a hash algorithm to randomly select forwarding paths for traffic, balancing traffic to multiple parallel paths to reduce 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

☆ This work was supported by the National Natural Science Foundation of China under Grant 62472050, Grant 62473146, Grant 62102046 and Grant 62072056; and the Natural Science Foundation of Hunan Province, China under Grant 2024JJ3017; and the open research fund of Key Lab of Broadband Wireless Communication and Sensor Network Technology (Nanjing University of Posts and Telecommunications), Ministry of Education under Grant JZNY202308; and the New Generation Information Technology Innovation Project 2023 under Grant 2023IT271; and the Science and Technology Project of Hunan Provincial Department of Water Resources under the Grant XSKJ2024064-36. A preliminary version of this paper appears in ICA3PP [1], TianJin, China, October, 2023.

* Correspondence to: Sanya Institute of Hunan University of Science and Technology, Sanya 572024, China.
E-mail address: jinwang@hnust.edu.cn (J. Wang).

https://doi.org/10.1016/j.comnet.2024.110867
Received 8 June 2024; Received in revised form 28 August 2024; Accepted 15 October 2024
Available online 23 October 2024

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] divides traffic into flowlets by detecting the time interval between packet clusters and randomly forwards them to different paths. 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 path.

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]. Moreover, coarse-grained schemes often result in low link utilization. Fine-grained packet-based schemes can effectively utilize multiple parallel paths, but they are prone to packet reordering issues. The switching granularity of flowlet schemes is between the above two categories, which can reduce the reordering of data packets while taking into account the link utilization. However, how to reasonably divide the traffic into flowlets and forward them to the appropriate path is still a problem worth studying [15,24].

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 [25–27]. 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 designed a flowlet-granularity rerouting mechanism based on dynamic timeout, which promptly reroutes traffic while avoiding packet reordering. Our focus is on distinguishing traffic that genuinely requires rerouting and minimizing the reordering issues caused by rerouting operations to the greatest extent possible [15,16]. In addition, we conduct performance comparisons between symmetric and asymmetric network topology to evaluate the performance of TDLB.

The main contributions of this paper are as follows:

- We conduct in-depth research and analysis on two issues during path congestion, including the increased tail latency caused by undifferentiated traffic that truly needs to be rerouted, and the longer flow completion time caused by reordering resulting from rerouting.
- We determine the path status by evaluating the queue length of the local switch and design a traffic recognition mechanism to differentiate between flows that contribute to path congestion and flows that truly need to be rerouted.

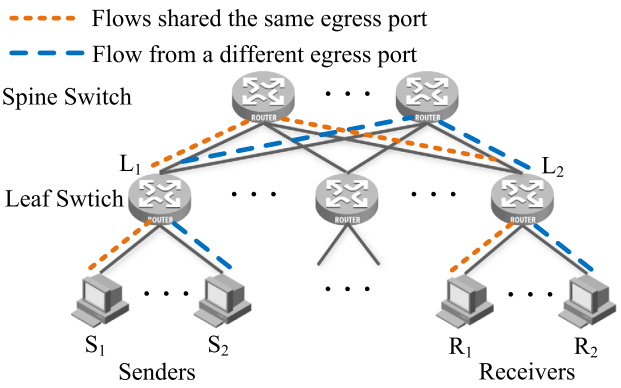


Fig. 1. Leaf-Spine topology.

- We propose a fine-grained load balancing mechanism TDLB that effectively separates traffic on congested paths and routes traffic with dynamic flowlet timeout to avoid packet reordering.
- We implement TDLB using NS-2 simulation, which effectively improves transmission performance under congested conditions and significantly reduces tail latency and average flow completion time (AFCT).

The rest of the paper is organized as followings. We set up the motivation in Section 2. In Section 3, we provide a detailed overview of the design and its details. In Section 4, we analyze the issue of packet reordering in asymmetric topology and propose solutions to address it. In Section 5, we conduct experimental simulations and analysis, including the analysis of relevant comparative results. In Section 6, we introduce some representative related work. In the final section, we summarize the work of this paper.

2. Motivation

However, current load balancing mechanisms have a common problem in that they all reroute all flows on congested paths without distinguishing between those that truly need to be rerouted. Firstly, the flows that truly need to be rerouted may still choose the same path as the flows that cause congestion, leading to an increase in tail latency. Secondly, rerouting increases the chance of reordering, which not only fails to reduce the flow completion time of congested flows but also leads to longer FCT [28,29].

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 L_1 and L_2 are connected to two sending terminals (S_1, S_2) and two receiving terminals (R_1, R_2), respectively. The two leaf switches are fully connected to two spine switches, meaning there are two identical parallel paths between L_1 and L_2, which have the same link bandwidth and latency. L_1 sends four flows ($f_1 \sim f_4$) to R_1 at R_2 moment, and S_2 subsequently sends a flow f_5 at T_2 moment. We use five typical load balancing mechanisms to handle this scenario.

As depicted in Fig. 2, whether it is ECMP at the flow level, LetFlow and CONGA at the flowlet level, or DRILL and Hermes at the packet level, they may all be directed to the same path when handling flows ($f_1 \sim f_5$) on leaf switch L_1. Moreover, a significant number of flows become congested in the queues of L_2 switches, with 80% of the flows being sent to the receiving terminal R_1 and the remaining 20% to the receiving terminal R_2. Consequently, this link experiences congestion, prompting the aforementioned load balancing mechanisms to redirect these flows and packets to the upper-level switch L_1, employing different processing schemes to reroute this portion of flows.

We analyze the impact of congestion on the completion time and tail latency of short flows through NS-2 simulation testing [26,30]. We use

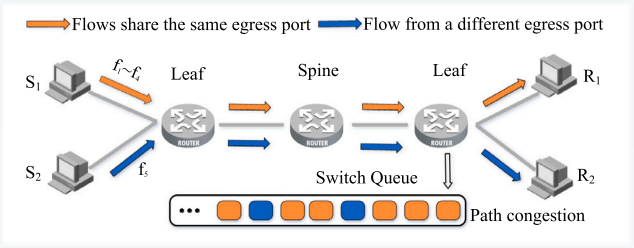


Fig. 2. Motivation example.

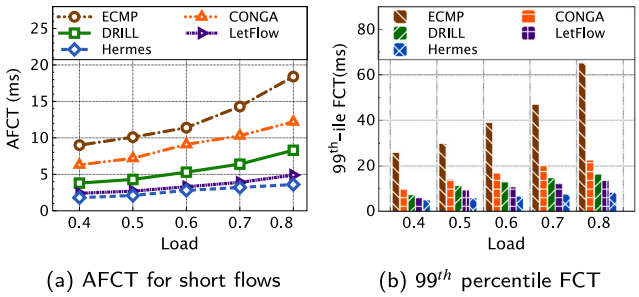


Fig. 3. Performance under different workloads.

a leaf-spine topology with three equal-cost paths between host pairs. The bottleneck bandwidth is 40 Gbps, and the round-trip propagation latency is 80 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.

A large number of flows ($f_1 \sim f_4$) with the same outbound port and a small number of flows such as f_5 with different outbound ports are crowded on the same transmission path, where flows ($f_1 \sim f_4$) are sent from leaf switch L_2 to receiver R_1, while flow f_5 is sent from leaf switch L_2 to receiver R_2. Obviously, it may lead to network performance bottlenecks and increase transmission latency. This is because a large amount of flows shares the same transmission path, which may not be able to handle all flows simultaneously. If not processed, a small amount of flows f_5 sent to different receiving ports will be blocked in the output queue of leaf switch L_2, seriously increasing the FCT of f_5. Therefore, a load balancing mechanism needs to be designed to deal with this situation.

Large tail 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 flows is rerouted to other parallel paths to reduce flow completion time and improve link utilization. As shown in Fig. 3a, as the network load increases, AFCT continues to increase. Fig. 3b shows the 99th percentile FCT for different load balancing mechanisms as network load increases. We observed that the tail 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. 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 tail latency of the flow.

Large average latency: The completion time of the flow actually increases. In Fig. 2, the flows ($f_1 \sim f_4$) sent to the same receiving terminal experience congestion on both the receiving terminal and the last hop switch. Fig. 3a shows the average flow completion time under different load balancing mechanisms as the load increases. It is worth noting that the flow completion time of ECMP is significantly higher than that of other methods. This is because ECMP uses a hash method and has no congestion awareness mechanism, but simply spreads traffic to different paths for forwarding. Even when other mechanisms reroute the flow from congested paths, the overall average flow completion time has significantly increased. Because flows that are rerouted and redirected to another path may still face congestion, increasing the possibility of disorderly transmission [6,14]. The flow that causes congestion cannot blindly reduce the FCT by rerouting. Disorderly arrival will actually lead to an increase in 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 flows, 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. In addition, the congested flow may generate new congestion after rerouting, and the only way to alleviate the congestion is to limit the sending rate, resulting in a decrease in overall throughput. These conclusions drive us to propose a congestion-aware load balancing mechanism with fine-grained that distinguishes between flows that genuinely require rerouting and flows that cause congestion, 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 tail latency. On the other hand, to prevent the flows that cause 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.

(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. Subsequently, the switch sends link congestion notifications to prevent 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.

(2) **Traffic separation:** The traffic separator is employed to separate congested traffic and other innocent traffic on congested paths, and to forward 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. TDLB employs the traffic separator only when congestion occurs in the link 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, reroute it to better transmission paths to enhance its performance. 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.

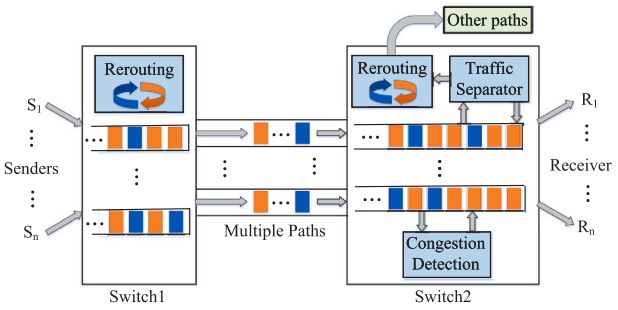


Fig. 4. TDLB overview.

3.2. Design rationale

Existing load balancing mechanism, there are no distinguishing flows on the congested path that truly need to be rerouted. They are forwarded to other parallel paths in the same method, resulting in increasing congestion and tail 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 tail 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 reroute 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 explain in detail the key design of each part of TDLB to explain how TDLB effectively separates traffic on congested paths and improves their network transmission performance.

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 [31,32]. 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 [29,33,34]. 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 flows from the innocent traffic.

Algorithm 1: Distinguishing congested flows

```
Input: A packet belongs to flow f
Output: Congested flow tag T
if cur_queue_length >queue_threshold and cur_path_delay
 >default_delay then
    Record current congested flows into the array;
    Mark congested flow tag T → True ;
end
for every packet do
    Assume the corresponding flow is f;
    if hash(f) exists in array then
        Mark congested flow tag T → True ;
    else
        Mark congested flow tag T → False
    end
    return T
end
```

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 local 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 congested flow tag has been added to the header of the data packet.

The mechanism for distinguishing congested flows is shown in Algorithm 1. When the local switch detects path congestion, the hash array will treat all recorded traffic as flow which causes the congestion (lines 3–7). After other subsequent traffic through hash operations, check if the hash value exists in the array (lines 8–10). If it does, mark the data packet with congestion, and increase the count corresponding to the hash value by one (lines 10–11); Otherwise, it would be considered innocent flows that truly need to be rerouted (lines 12–13).

Truly necessary rerouting: The rerouting rules play a crucial role in the performance of load balancing mechanisms [35–37]. 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 TDLB employs flowlet granularity for traffic rerouting, determining the forwarding path based on path delay to route traffic that truly requires rerouting to other improved paths. Specifically, when data packets enter the switch, detected innocent traffic triggers the rerouting mechanism to forward it to other enhanced parallel paths. At this point, the switch detects the time intervals between packets, rerouting flowlets with gaps larger than the flowlet timeout to better parallel paths. For the selection of parallel paths, it is determined based on path delay, scanning to find the path with the lowest link delay as the rerouting path. Through necessary rerouting, innocent traffic performance can be effectively enhanced, packet reordering avoided, and congestion on the original path alleviated and eliminated. We will further describe this design in Section 4. For congested flows, their sending rates are limited through congestion control mechanisms, thereby alleviating path congestion as early as possible.

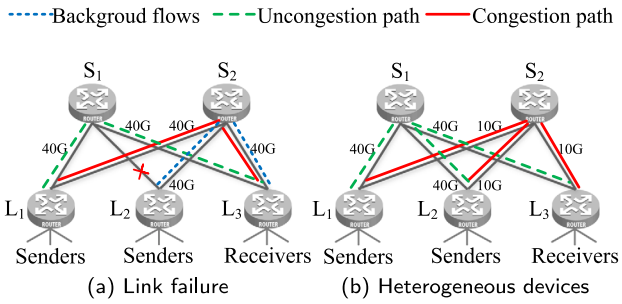


Fig. 5. Asymmetry topology.

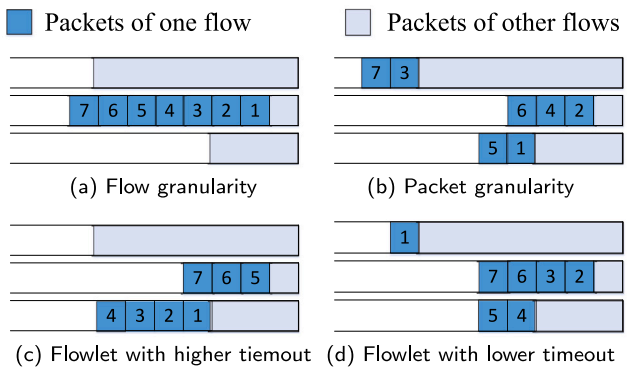


Fig. 6. Examples of different granularities.

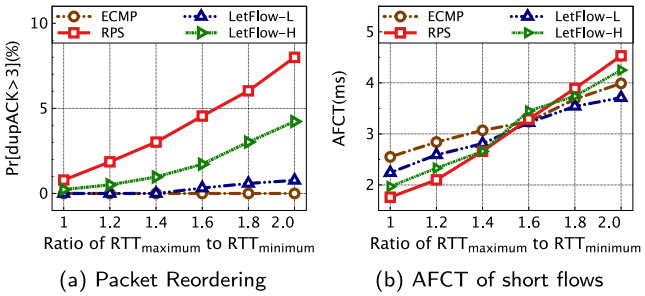


Fig. 7. The impact of asymmetry on different granularities.

4. Prevent packet reordering

In this section, we will explore the potential issues of different granularity load balancing mechanisms in asymmetric topology and the fixed flowlet timeout method at the flowlet granularity level, and propose corresponding solutions to enhance the robustness of TDLB in preventing packet reordering.

4.1. Problem analysis

Asymmetric topology: Modern DCNs support the simultaneous transmission of traffic across multiple paths between hosts. Due to reasons such as dynamic traffic, link failures and coexistence of heterogeneous devices, significant differences in latency between paths may occur, leading to common occurrences of asymmetric topology in DCNs [15].

As shown in Fig. 5a, a link failure has occurred between leaf switch L_2 and spine switch S_1. The traffic sent from L_2 to L_3 can only be routed through L_2-S_2-S_3, causing the path from S_2 to L_3 to become heavily congested. Consequently, the two paths from L_1 to L_3 become asymmetrical, with traffic passing through S_2 experiencing greater link latency than traffic passing through S_1. Fig. 5b illustrates the topology asymmetry caused by the coexistence of heterogeneous devices in a data center network. Spine switch S_1 supports a bandwidth of 40 Gbps, while S_2 only supports 10 Gbps. This also results in two paths with different latencies between L_1 and L_3 or L_2 and L_3.

Load Balancing with different granularities: Typically, existing load balancing schemes operate at the granularity of flow, flowlet, and packet. Due to more significant latency differences between paths in asymmetric topology, fine-grained load balancing can lead to more severe packet reordering. In Fig. 6, three available parallel paths are provided for traffic, and traffic rerouting is performed through different granularities of switching. Meanwhile, for the flowlet-based approach, analyze the situations that may occur when setting a higher and a lower timeout, respectively, and explain the differences between them. Fig. 6a illustrates that flow-based schemes do not cause packet reordering. But flows on congested paths have no opportunity for rerouting. Packet-based schemes, as shown in Fig. 6b, cause serious packet reordering. Flowlet is defined as a burst of packets within a flow [14]. A new flowlet is identified when the inter-arrival time between adjacent packets exceeds a certain timeout, thereby adjusting the forwarding strategy for the new flowlet. From Fig. 6c and d we can see that different flowlet timeout settings result in different degrees of packet reordering. On the one hand, flowlet schemes with a larger timeout may miss opportunities for rerouting, failing to alleviate congestion on the original path promptly. On the other hand, setting a smaller flowlet timeout will result in even more severe packet reordering.

We analyze the performance impact of changes in the degree of asymmetry on coarse-grained and fine-grained load balancing schemes. We utilize a Leaf-Spine topology with three equivalent paths between host pairs. To evaluate the influence of the degree of asymmetry, we introduce an additional delay to one of the paths. Among the three paths, the ratio of maximum RTT to minimum RTT vary between 1 to 2. We compare the performance of ECMP, RPS, Letflow-H and Letflow-L, where Letflow-H represents a higher timeout and Letflow-L represents a lower timeout, which are set to 500 μs and 100 μs respectively.

Fig. 7a illustrates the proportion of 3-dupack events caused by packet reordering relative to all packets. ECMP does not induce packet reordering. LetFlow-H also encounters minimal packet reordering issues. However, LetFlow-L and RPS experience a significant increase in packet reordering as the degree of topology asymmetry increases. Fig. 7b presents the AFCT. When the degree of asymmetry is low, RPS achieves optimal performance due to its effective utilization of multiple paths. Additionally, Letflow-L outperforms Letflow-H and ECMP due to more rerouting opportunities. As the degree of asymmetry increases, the impact of packet reordering on performance becomes more pronounced. In such scenarios, Letflow-H and ECMP exhibit better performance compared to Letflow-L.

4.2. How to set timeout dynamically

Then we demonstrate how TDLB dynamically adjusts flowlet timeout. We track the latency differences between multiple paths connecting pairs of servers and Top-of-Rack (ToR) switches. Based on the dynamic latency differences, a dynamic flowlet timeout is computed for each pair of ToR switches. The challenge we face is how to track the latency differences between congested paths and the fastest path, as well as how to calculate an appropriate flowlet timeout.

Tracking latency of different paths: To track the latency differences between congested paths and the fastest path, TDLB instructs each leaf switch to periodically send probe packets to upper-layer switches at a specific frequency, traversing all available paths. In a Leaf-Spine topology, spine switches add their local switch IDs to the probe packets and forward them to other leaf switches. In a three-tier topology, when aggregation switches connected to the leaf switch sending probe packets receive them, they need to forward the packets

Algorithm 2: Timeout update

```
Input: A probe packet
Output: Updated delay table and timeout table
path = Hash(switch_ids);
delay = arrival_time - send_time;
if path == min_path then
    min_delay = delay;
else if math == max_path then
    max_delay = delay;
else
    if delay < min_delay then
        min_delay = delay;
        min_path = path;
    else if delay > max_delay then
        max_delay = delay;
        max_path = path;
    end
end
if min_delay or max_delay changed then
    timeout = max_delay - min_delay;
    if min_delay changed then
        Update next_hop for fastest path;
    end
end
```

both to upper-layer switches and to other leaf switches connected to them. However, other aggregation switches only need to forward probe packets to lower-layer switches to prevent unnecessary loop transmissions [38].

The leaf switches receiving probe packets need to determine the path ID and its latency based on the information contained in the probe packets. Therefore, the probe packets will include the following information: the ID of the leaf switch sending the probe packet, the IDs of intermediate switches the probe packet traverses, and a timestamp field indicating when the probe packet is sent. In a leaf-spine topology, there is only one hop between ToR pairs, so at most, it requires carrying two 24-bit switch IDs and one 32-bit timestamp, totaling 80 bits of information. In a three-tier topology, there can be up to three hops between ToR pairs, requiring carrying up to four switch IDs and one timestamp, totaling 108 bits. Therefore, including the normal Ethernet and IP headers, a minimum-sized 64-byte packet is sufficient to meet the requirements.

Dynamically adjust flowlet timeout: Most flowlet-based load balancing schemes statically set the timeout to a fixed value greater than the maximum delay difference between paths. The rationale behind this approach is to avoid packet reordering, which is indeed a valid concern and should also be adhered to when dynamically setting the timeout. However, it is important to note that the maximum delay difference between paths varies both temporally and spatially. Firstly, on a temporal scale, the maximum delay difference between ToR switch pairs fluctuates due to the highly dynamic nature of traffic intensity in the network. Secondly, on a spatial scale, significant differences in the maximum delay difference between different ToR switch pairs can occur, especially in asymmetric topology. Therefore, it is necessary to dynamically set a flowlet timeout for each ToR switch pair.

We calculate the flowlet timeout based on the latency difference between the fastest and slowest paths between each pair of ToR switches. Thus, we maintain a delay table on each ToR switch, which includes the latency of the fastest and slowest paths along with their corresponding hash values. Additionally, we maintain a timeout table to dynamically update the flowlet timeout for each ToR switch pair. We can also include information about the shortest path (such as the next hop) in the timeout table to select the best path for rerouting flowlets. When a probe packet enters a ToR switch, the switch performs the following operations to update the information in the delay table and timeout table.

The mechanism for timeout update is shown in Algorithm 2. When the switch receives a probe packet, the timeout update algorithm first computes the hash value of the path (line 3) the delay of the packet (line 4). Then, if the path is the same as the fastest path or slowest path recorded in the delay table, update the minimum or maximum delay corresponding to the path (lines 5–8). Otherwise, the delay of the path is compared with the minimum delay and the maximum delay to update the delay table (lines 9–17). If the minimum or maximum delay changes, it updates the timeout and the next-hop information for the fastest path (lines 18–23), dynamically optimizing network path selection.

By dynamically adjusting the flowlet timeout between each pair of ToR switches, switches can flexibly partition flow into flowlets based on the latency conditions across multiple paths. In scenarios where the latency difference between paths is minimal, congested traffic can be forwarded to better paths in a timely manner with a smaller flowlet timeout. In cases where asymmetric topology results in significant latency differences between paths, a larger flowlet timeout can prevent packet reordering caused by rerouting operations.

5. Evaluation

In this section, we use large-scale NS-2 simulations to evaluate the performance of TDLB and compare it with state-of-the-art load balancing mechanisms to illustrate the superiority of TDLB.

Baseline: 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 multiple path. 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.

Workload: We conduct experiments with two classic workloads, Web Search and Data Mining, to evaluate the performance of TDLB in large-scale scenarios [6,7]. In general, flows smaller than 100 KB 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 1 MB, while in Data Mining workload, less than 5% of traffic is greater than 35 MB. Web Search and Data Mining have 62% and 83% of short flows smaller than 100 KB, respectively, resulting in a heavy-tailed distribution.

Topology: 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 40 Gbps bandwidth link, with a round-trip propagation latency of 80 μs. The traffic in the network is generated randomly through the Poisson distribution between random host pairs. We adjust the workload from 0.4 to 0.8 to thoroughly evaluate the performance of TDLB.

5.1. Performance under symmetric topology

In this section, we test the performance of TDLB under symmetric network topology.

Fig. 8 shows the AFCT of short flows under Web Search and Data Mining workloads, respectively. Compared with the other five load balancing mechanisms, TDLB significantly reduces AFCT, especially

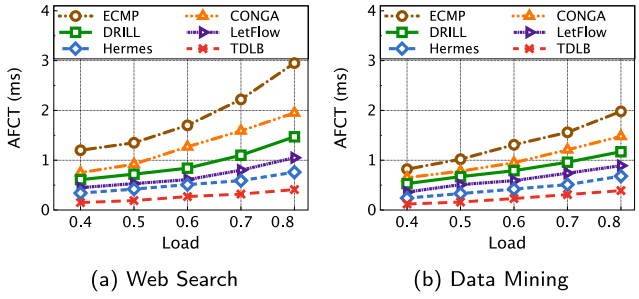


Fig. 8. AFCT in different workloads under symmetric topology.

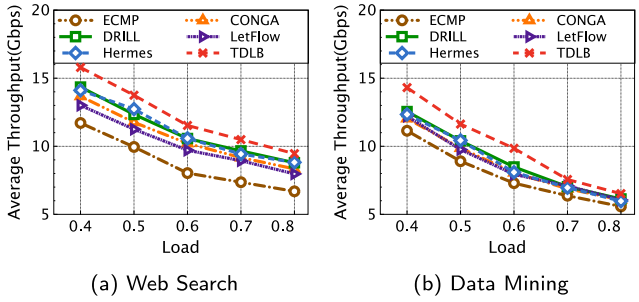


Fig. 9. Throughput in different workloads under symmetric topology.

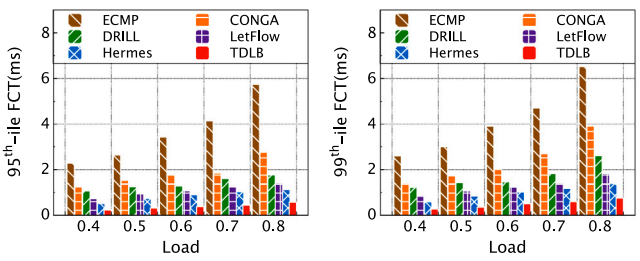


Fig. 10. Tail latency in Web Search under symmetric topology.

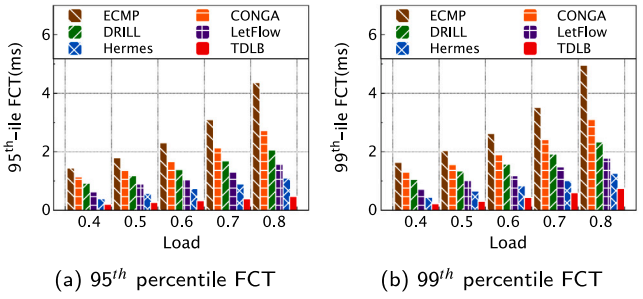


Fig. 11. Tail latency in Data Mining under symmetric topology.

in high workload situations. These test results indicate that TDLB performs well in low latency transmission. 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.

TDLB 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. 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. For example, under a 0.7 workload, in the Web Search scenario, TDLB decreases by ~86%, ~80%, ~71%, ~60%, and ~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 ~80%, ~74%, ~68%, ~58%, and ~39%, respectively.

However, long flows are not sensitive to latency but are more concerned with the throughput of the link during transmission. Therefore, we compared the throughput of TDLB with five load balancing schemes under different loads. Fig. 9 shows the throughput of long flows under Web Search and Data Mining workloads, respectively. As the workload increases, the throughput of long flows gradually decreases across all schemes. However, compared to the other five solutions, TDLB significantly improves the throughput of long flows. This is because TDLB distinguishes between flows that contribute to congestion and flows that truly need to be rerouted. On one hand, congested flows remain on their original paths instead of being rerouted to other paths, thereby minimizing the impact on the transmission rate by avoiding new congestion. On the other hand, innocent flows are cautiously rerouted to better paths at an appropriate granularity without causing packet reordering, thus preventing the throughput loss caused by packet reordering.

Next, we test the 95th percentile FCT and 99th percentile FCT of short flows in different workloads as shown in Figs. 10 and 11. Obviously, compared to the other five solutions, TDLB significantly improves the tail FCT, especially in high workload situations. Severe tail latency 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 selects forwarding paths, without considering the congestion situation of other parallel paths, and transmits at the flow level. Therefore, ECMP has the greatest tail latency. 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. Specifically, under a 0.7 workload, in the Web Search scenario, TDLB decreases by ~89%, ~76%, ~73%, ~65%, and ~49% 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 ~83%, ~75%, ~69%, ~59%, and ~41%, respectively.

Furthermore, we are surprised to find that under the same workload, the 99th percentile FCT is approximately twice that of AFCT. We conduct a detailed theoretical analysis of the experimental results. Firstly, the 99th percentile FCT represents the portion of the flow with the highest tail 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 tail latency includes the transmission latency before and after retransmission, which is approximately twice that of AFCT.

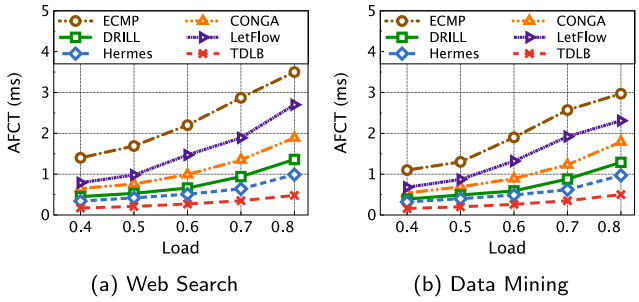


Fig. 12. AFCT in different workloads under asymmetric topology.

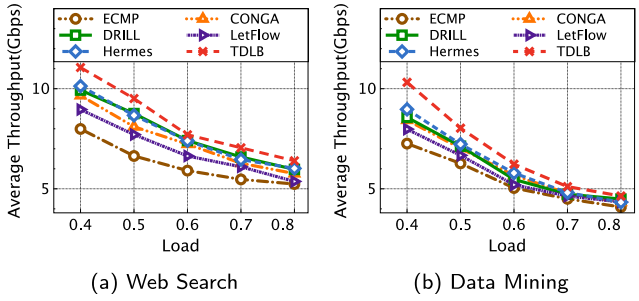


Fig. 13. Throughput in different workloads under asymmetric topology.

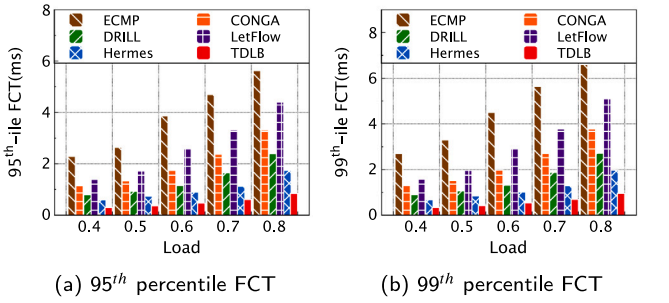


Fig. 14. Tail latency in Web Search under asymmetric topology.

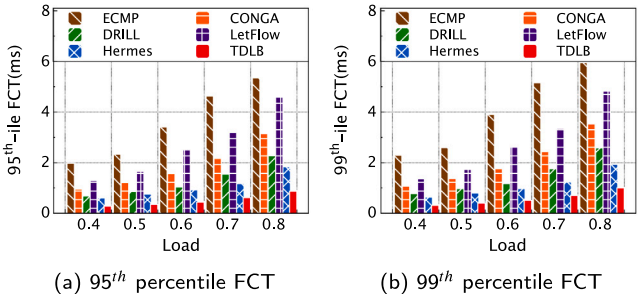


Fig. 15. Tail latency in Data Mining under asymmetric topology.

5.2. Performance under asymmetric topology

In this section, we test the performance of TDLB under asymmetric network topology. The experimental setup in the asymmetric scenario is basically the same as that in the symmetric scenario. To generate asymmetry, we randomly select a path where a spine switch is located and modify its bandwidth from 40 Gbps to 10 Gbps.

Overall, TDLB can also exhibit excellent performance in asymmetric topology. Fig. 12 shows the AFCT in 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. As shown in Fig. 13, the throughput of long flows decreases as the load increases. However, it is evident that TDLB still performs better than other schemes, especially when the load is relatively low. Figs. 14 and 15 respectively show the 95th percentile FCT and 99th percentile FCT in different workloads with asymmetric topology. Compared with other schemes, TDLB still has the lowest tail latency at both high and low loads.

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 [39–42]. Due 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 is 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 and throughput.

In addition, they do not distinguish the traffic that needs to be rerouted before rerouting, which can cause congestion when the traffic that causes congested 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 flows continue to queue on the original path, replacing rerouting with speed reduction to avoid disorder and significantly reduce the FCT of the congested flows. On the other hand, TDLB dynamically adjusts the flowlet timeout based on the latency differences between paths, thereby rerouting traffic to the best path without causing packet reordering. In cases of topological asymmetry, TDLB is better able to adapt to the latency differences between paths and make optimal routing decisions. Therefore, TDLB can better meet the transmission requirements of low latency and high throughput.

6. Related work

Existing load balancing mechanisms can be divided into different categories according to transmission granularity, which we summarize as follows:

Flow-based Rerouting Granularity. ECMP [11] achieves load balancing by distributing network traffic across multiple path with equal costs. 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.

Packet-based Rerouting Granularity. RPS [16] offers a straightforward packet-level load balancing mechanism by randomly distributing packets among all available transmission paths on the switch. 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. 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.

Flowlet-based Rerouting Granularity. 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 conditions. 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.

The above load balancing solutions work well in certain network scenarios, but none of them considers distinguishing flows that really need to be rerouted. In contrast, TDLB distinguishes different flows and performs differentiated processing. In addition, TDLB also performs fine-grained routing with dynamic flowlet timeout, and works well in different network scenarios. In the future, how to use TDLB in a real network testbed is a problem we will delve into.

7. Conclusion

This paper presents TDLB, a fine-grained load balancing mechanism to solve the problem of large tailing latency caused by coarse-grained rerouting. 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 flows that truly need rerouting will dynamically adjust the flowlet timeout based on the latency differences between paths, flexibly rerouting to the best path. The experimental results of NS-2 simulation show that compared with existing mechanisms such as Hermes in symmetrical scenes, TDLB can decrease the tail latency and AFCT by up to 49% and 46%, respectively.

CRediT authorship contribution statement

Jinbin Hu: Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Ruiqian Li:** Writing – original draft, Visualization, Validation, Software, Resources, Methodology. **Ying Liu:** Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology. **Jin Wang:** Writing – review & editing, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

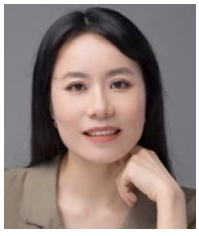
Data availability

No data was used for the research described in the article.

References

[1] J. Hu, Y. Liu, S. Rao, J. Wang, D. Zhang, Enabling traffic-differentiated load balancing for datacenter networks, in: Proc. ICA3PP, 2023, pp. 250–269.
[2] Y. Wang, Y. Li, T. Wang, G. Liu, Towards an energy-efficient Data Center Network based on deep reinforcement learning, Comput. Netw. 210 (2022) 108939.
[3] W. Lyu, J. Huang, J. Liu, Z. Li, S. Zou, W. Li, J. Wang, D. Zhang, Mitigating port starvation for shallow-buffered switches in datacenter networks, in: Proc. IEEE ICDCS, 2021, pp. 921–931.
[4] J. Hu, C. Zeng, Z. Wang, J. Zhang, K. Guo, H. Xu, J. Huang, K. Chen, Load balancing with multi-level signals for lossless datacenter networks, IEEE/ACM Trans. Netw. (2024) 1–13.
[5] J. Huang, W. Lyu, W. Li, J. Wang, T. He, Mitigating packet reordering for random packet spraying in data center networks, IEEE/ACM Trans. Netw. 29 (3) (2021) 1183–1196.
[6] J. Wang, S. Rao, Y. Liu, P.K. Sharma, J. Hu, Load balancing for heterogeneous traffic in datacenter networks, J. Netw. Comput. Appl. 217 (2023) 103692.
[7] J. Hu, C. Zeng, Z. Wang, J. Zhang, K. Guo, H. Xu, J. Huang, K. Chen, Enabling load balancing for lossless datacenters, in: Proc. IEEE ICNP, 2023, pp. 1–11.
[8] J. Hu, H. Shen, X. Liu, J. Wang, Rdma transports in datacenter networks: survey, in: Proc. IEEE/ACM IWQOS, IEEE Netw. (2023) 1–8.
[9] W. Li, X. Yuan, K. Li, H. Qi, X. Zhou, Leveraging endpoint flexibility when scheduling coflows across geo-distributed datacenters, in: Proc. IEEE INFOCOM, 2018, pp. 873–881.
[10] W. Bai, K. Chen, S. Hu, K. Tan, Y. Xiong, Congestion control for high-speed extremely shallow-buffered datacenter networks, in: Proc. ACM APNet, 2017, pp. 29–35.
[11] C. Hopps, Analysis of an equal-cost multi-path algorithm, 2000.
[12] M. Alizadeh, T. Edsall, et al., CONGA: Distributed congestion-aware load balancing for datacenters, in: Proc. ACM SIGCOMM, 2014, pp. 503–514.
[13] S. Ghorbani, Z. Yang, P.B. Godfrey, Y. Ganjali, A. Firoozshahian, DRILL: Micro load balancing for low-latency data center networks, in: Proc. ACM SIGCOMM, 2017, pp. 225–238.
[14] E. Vanini, R. Pan, M. Alizadeh, P. Taheri, T. Edsall, Let it flow: Resilient asymmetric load balancing with flowlet switching, in: Proc. USENIX NSDI, 2017, pp. 407–420.
[15] H. Zhang, J. Zhang, W. Bai, K. Chen, M. Chowdhury, Resilient datacenter load balancing in the wild, in: Proc. ACM SIGCOMM, 2017, pp. 253–266.
[16] A. Dixit, P. Prakash, Y.C. Hu, R.R. Kompella, On the impact of packet spraying in data center networks, in: Proc. IEEE INFOCOM, 2013, pp. 2130–2138.
[17] J. Hu, J. Huang, Z. Li, J. Wang, T. He, A receiver-driven transport protocol with high link utilization using anti-ECN marking in data center networks, IEEE Trans. Netw. Serv. Manag. 20 (2) (2023) 1898–1912.
[18] X. He, W. Li, S. Zhang, K. Li, Efficient control of unscheduled packets for credit-based proactive transport, in: Proc. IEEE ICPADS, 2023, pp. 593–600.
[19] A. Kabbani, B. Vamanan, J. Hasan, F. Duchene, FlowBender: Flow-level adaptive routing for improved latency and throughput in datacenter networks, in: Proc. ACM CoNEXT, 2014, pp. 149–160.
[20] J. Wang, D. Yuan, W. Luo, S. Rao, R.S. Sherratt, J. Hu, Congestion control using in-network telemetry for lossless datacenters, Comput. Mater. Continua 75 (1) (2023) 1195–1212.
[21] K. Wen, Z. Qian, S. Zhang, S. Lu, OmniFlow: Coupling load balancing with flow control in datacenter networks, in: Proc. IEEE ICDCS, 2016, pp. 725–726.
[22] M. Shafiee, J. Ghaderi, A simple congestion-aware algorithm for load balancing in datacenter networks, in: Proc. IEEE INFOCOM, 2016, pp. 1–9.
[23] M. Alizadeh, A. Greenberg, D.A. Maltz, et al., Data center TCP (DCTCP), in: Proc. ACM SIGCOMM, 2010, pp. 63–74.
[24] J. Liu, J. Huang, W. Li, J. Wang, AG: Adaptive switching granularity for load balancing with asymmetric topology in data center network, in: Proc. IEEE ICNP, 2019, pp. 1–11.
[25] A. Munir, I.A. Qazi, Z.A. Uzmi, A. Mushtaq, S.N. Ismail, M.S. Iqbal, B. Khan, Minimizing flow completion times in data centers, in: Proc. IEEE INFOCOM, 2013, pp. 2157–2165.
[26] Z. Li, W. Bai, K. Chen, D. Han, Y. Zhang, D. Li, H. Yu, Rate-aware flow scheduling for commodity data center networks, in: Proc. IEEE INFOCOM, 2017, pp. 1–9.
[27] D. Zats, T. Das, P. Mohan, D. Borthakur, R. Katz, DeTail: Reducing the flow completion time tail in datacenter networks, in: Proc. ACM SIGCOMM, 2012, pp. 139–150.
[28] T. Benson, A. Akella, D.A. Maltz, Network traffific characteristics of data centers in the wild, in: Proc. ACM IMC, 2010, pp. 267–280.
[29] C. Hu, B. Liu, H. Zhao, K. Chen, Y. Chen, Y. Cheng, H. Wu, Discount counting for fast flow statistics on flow size and flow volume, IEEE/ACM Trans. Netw. 22 (3) (2013) 970–981.
[30] The NS-2 network simulator. http://www.isi.edu/nsnam/ns.

[31] W. Bai, S. Hu, K. Chen, K. Tan, Y. Xiong, One more config is enough: Saving (DC) TCP for high-speed extremely shallow-buffered datacenters, IEEE/ACM Trans. Netw. 9 (2) (2020) 489–502.
[32] Z. Liu, K. Chen, H. Wu, S. Hu, Y.C. Hut, Y. Wang, G. Zhang, Enabling work-conserving bandwidth guarantees for multi-tenant datacenters via dynamic tenant-queue binding, in: Proc. IEEE INFOCOM, 2018, pp. 1–9.
[33] C. Hu, B. Liu, H. Zhao, K. Chen, Y. Chen, C. Wu, Y. Cheng, Disco: Memory efficient and accurate flow statistics for network measurement, in: Proc. IEEE ICDCS, 2010, pp. 665–674.
[34] W. Wei, H. Gu, W. Deng, Z. Xiao, X. Ren, ABL-TC: A lightweight design for network traffic classification empowered by deep learning, Neurocomputing 489 (2022) 333–344.
[35] W. Wei, L. Fu, H. Gu, Y. Zhang, T. Zou, C. Wang, N. Wang, GRL-PS: Graph embedding-based DRL approach for adaptive path selection, IEEE Trans. Netw. Serv. Manag. 20 (3) (2023) 2639–2651.
[36] J. Hu, Y. He, J. Wang, W. Luo, J. Huang, RLB: Reordering-robust load balancing in lossless datacenter network, in: Proc. ACM ICPP, 2023, pp. 576–584.
[37] J. Hu, Y. He, W. Luo, J. Huang, J. Wang, Enhancing load balancing with in-network recirculation to prevent packet reordering in lossless data centers, IEEE/ACM Trans. Netw. (2024) 1–14.
[38] N. Katta, M. Hira, C. Kim, A. Sivaraman, J. Rexford, HULA: Scalable load balancing using programmable data planes, in: Proc. ACM SOSR, 2016, pp. 1–12.
[39] J. Hu, J. Huang, J. Wang, J. Wang, A transmission control mechanism for lossless datacenter network based on direct congestion notification, Acta Electron. Sin. 51 (9) (2023) 2355–2365.
[40] J. Zheng, Z. Du, Z. Zha, Z. Yang, X. Gao, G. Chen, Learning to configure converters in hybrid switching data center networks, IEEE/ACM Trans. Netw. 32 (1) (2024) 520–534.
[41] Y. Liu, W. Li, W. Qu, H. Qi, BULB: Lightweight and automated load balancing for fast datacenter networks, in: Proc. ACM ICPP, 2022, pp. 1–11.
[42] Y. Zhao, Y. Huang, K. Chen, M. Yu, S. Wang, D.S. Li, Joint VM placement and topology optimization for traffic scalability in dynamic datacenter networks, Comput. Netw. 80 (2015) 109–123.



Jinbin Hu received the B.E. and M.E. degrees from Beijing Jiao Tong University, China, in 2008 and 2011, respectively, and the Ph.D. degree in computer science from Central South University, China, in 2020. She is currently an associate professor at Changsha University of Science and Technology and a Post-Doc in Hong Kong University of Science and Technology. Her current research interests are in the area of datacenter networks, RDMA networking and learning-basednetwork systems.

Ruiqian Li received the B.E. degree from Hohai University, China, in 2023. Now, he is currently pursuing the M.E. degree in the School of Computer and Communication Engineering at Changsha University of Science and Technology, China. His research interests include datacenter networks.

Ying Liu received the B.E. degree from Jiangxi Normal University, China, in 2022. Now, he is currently pursuing the M.E. degree in the School of Computer and Communication Engineering at Changsha University of Science and Technology, China. His research interests include datacenter networks and evolutionary computation.

Jin Wang received the B.S and M.S. degree from Nanjing University of Posts and Telecommunications, China in 2002, 2005 respectively. He received Ph.D. degree from Kyung Hee University Korea in 2010. Now, he is a professor at Hunan University of Science and Technology. He has published more than 400 international journal and conference papers. His research interests mainly include wireless ad hoc and sensor network, network performance analysis and optimization etc. He is a Fellow of IET and a senior member of the IEEE.