Hindawi
Security and Communication Networks
Volume 2022, Article ID 9594517, 17 pages
https://doi.org/10.1155/2022/9594517

WILEY | Hindawi

Research Article

Coarse-Grained Load Balancing with Traffic-Aware Marking in Data Center Networks

Ran Huang,[1,2,3] Jingliang Zhang,[1] Yuanzhen Hu,[1] Shaojun Zou,[1,2] Xidao Luan,[1] Jinbin Hu,[4] Chang Ruan,[4] and Tao Zhang[1,2]

[1]*School of Computer Engineering and Applied Mathematics, Changsha University, Changsha 410022, China*
[2]*Hunan Province Key Laboratory of Industrial Internet Technology and Security, Changsha University, Changsha 410022, China*
[3]*Department of Computer Science, The University of Sheffield, Sheffield S10 2TN, UK*
[4]*School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 4100114, China*

Correspondence should be addressed to Xidao Luan; lxd@ccsu.edu.cn

Received 30 April 2022; Accepted 7 October 2022; Published 19 October 2022

Academic Editor: Shahram Babaie

In the data center environment, meeting the dissimilar requirements of transmission performance for long and short data flows is crucial for guaranteeing the data center service quality. It is well known that optimizing the network transmission performance is the key. Abundant research studies have reported that designing an effective load-balancing scheme can boost the network performance by enhancing the path utilization of multirooted data center networks. Unfortunately, the existing data center load-balancing schemes fail to provide satisfying flow-level transmission performance since they choose to ignore the constant changing path status and blindly assign paths to data flows regardless of their different performance requirements. In order to solve these issues, we propose a data center load-balancing scheme called TAM to meet the performance requirements of long and short flows simultaneously. In detail, TAM leverages the traffic-aware ECN marking mechanism to force long flows to proactively maintain the switch queue length at a low level and release bandwidth to short flows when they compete for the bottleneck link. On the contrary, TAM carries out the coarse-grained routing or rerouting for short and long flows, respectively, in real time by perceiving the path status. In this way, the short flows achieve low delay transmission while the long flows obtain high throughput. The simulation results based on NS2 tests show that TAM achieves up to 60% lower FCTs for short flows and a 40% improvement in goodput for long flows.

1. Introduction

A modern data center usually hosts a variety of applications [1–4], mainly including latency-sensitive applications (e.g., search, RPCs, and gaming) and throughput-sensitive applications (such as video, big data analytics, and VM migration). Latency-sensitive applications commonly generate plenty of short flows, each of which has a small amount of data (less than 100 KB) to be transferred and needs the data center network (DCN) to provide it with low queuing delay [5–10]. Throughput-sensitive applications create a small number of long flows [8–11]. Each long flow possesses a large amount of data (more than 100 KB) and requires persistent throughput during data transmission. These heterogeneous flows are mixed in DCN [12]. To provide better service quality and user experience [13], DCN should meet their performance requirements at the same time. However, this challenge remains elusive currently.

Luckily, in recent advances, there is a promising method to address the above challenge. The topology of DCN (e.g., leaf-spine topology [14, 15]) often has multiple parallel end-to-end paths. Deploying an efficient load-balancing mechanism on DCN can fully utilize all these parallel paths, thus providing high network bisection bandwidth, which in turn boosts the network transmission performance [9, 10, 16]. Nonetheless, due to the inefficient path selection strategy in

existing data center load-balancing schemes, the proliferated end-to-end bandwidth has not yet brought prominent performance improvement as expected.

For example, as a widely deployed flow-level load-balancing mechanism, equal-cost multipath (ECMP) [17] uses a hash algorithm to distribute flows to all available paths, while its path utilization is still quite low because of suffering from the problem of hash collision. To fully utilize all the parallel paths, both random packet spraying (RPS) [18] and digit-reversal bouncing (DRB) [19] perform packet-level rerouting. However, they select a path for each packet either in a random or round-robin manner, thus leading to a serious packet reordering problem. LetFlow [20] determines whether to reroute the subsequent packets of current flow by judging whether a flowlet arises, and Presto [21] performs the round-robin rerouting for each flow cell. Although these two schemes strike a good balance between reducing packet reordering and maintaining high path utilization compared to the former strategies, they also do not perceive the path condition, so their rerouting operations are blind, causing the problem of congestion mismatch. Because these mechanisms do not perceive whether the data flow is a long flow or a short flow, they cannot guarantee their performance requirements according to the type of data flow on the bottleneck link.

Furthermore, the common problem of these load-balancing mechanisms is that they treat all kinds of heterogeneous data flows indiscriminately when making path selection and naturally do not perceive their different characteristics and transmission performance requirements. Consequently, short flows are forced to experience large queuing delays or choose high delay paths, while long flows have to struggle between how to improve path utilization and avoid packet reordering. To address these efficiencies, we design TAM to meet the performance requirements of both short and long flows simultaneously. Specifically, we designed a coarse-grained data center load-balancing scheme called TAM to improve the performance of short flows by carrying out the path-status perceived routing and the long flow dominated queue length controlling and bandwidth compromising. In the meantime, TAM monitors the load status and balance level across all parallel paths in real time to conduct flowlet-level rerouting for long flows, which greatly compensates for the performance loss from bandwidth compromising, thus guaranteeing the transmission performance of both long and short flows at the same time.

In summary, we make the following contributions:

(i) We conduct a large number of simulation experiments to verify the impact of long and short flows on each other's performances when they pass through the same bottleneck link.

(ii) We propose TAM to simultaneously improve the flow-level transmission performance of short and long flows. The TAM employs the traffic-aware ECN marking to force the long flows to proactively retreat, thus controlling the switch queue length at a very low level and helping the short flows completely quickly. Besides, TAM perceives the path status in real time and carries out flowlet-level rerouting for long flows to enhance the path utilization.

(iii) We constructed large-scale NS2 simulation tests to evaluate the performance of TAM. It can be seen from the results that compared with the representative data center load-balancing scheme, TAM reduces the AFCTs and Tail FCTs of the short flows by up to 60% and 50%, respectively, as well as increases the average goodput of long flows by up to 40%.

The rest of this study is organized as follows. In Section 1.1, we analyze the negative effects of the existing representative load-balancing mechanisms on short flow and long flow and then put forward our design objectives. In Section 2, we design TAM to meet the requirements of both short and long flows. In Section 4, we conduct different scale experiments based on the NS2 simulator to evaluate TAM's performance. After that, we discuss related work in Section 5 and summarize our work in Section 6.

1.1. Problem Description. In a data center, the long and short flows mix together and are transmitted on the same physical network [22, 23]. Their performance requirements are entirely different. The short flow needs to experience extremely low queuing delay, while the long flow requires high throughput transmission. However, the existing typical load-balancing mechanisms cannot meet their performance requirements at the same time. In Section 1.2, to elaborate on this issue and clearly illustrate how long and short flows interact, we choose ECMP, LetFlow, and RPS as representatives to analyze why the flow-level transmission performances of heterogeneous data flows cannot be simultaneously guaranteed. ECMP, LetFlow, and RPS have been widely used in industry and academia. The former two schemes are the representatives of coarse-grained mechanisms, while the latter one is the representative of fine-grained mechanisms.

As shown in Figure 1, ECMP is unaware of the status of the transmission paths when transmitting packets. This situation leads to two long flows mixed on the same path (path1) but makes no packets travel through paths 2 and 3. As a result, the transmission links become extremely unbalanced in load, and the transmission performance of the data flows decreases significantly. Meanwhile, the transmission completion time of a short flow is significantly increased since the switch queue length of path 1 is too large.

Under LetFlow, the short flow again experiences a large queuing delay since it is still with long flows on path 2. Meanwhile, the input traffic is also not scattered evenly, so the available path 3 is still unused.

With RPS, the input traffic is distributed to all the available paths evenly, but long flows experience serious packet reordering, which may seriously disturb the control logic of the TCP [24, 25] stack at the end host, causing

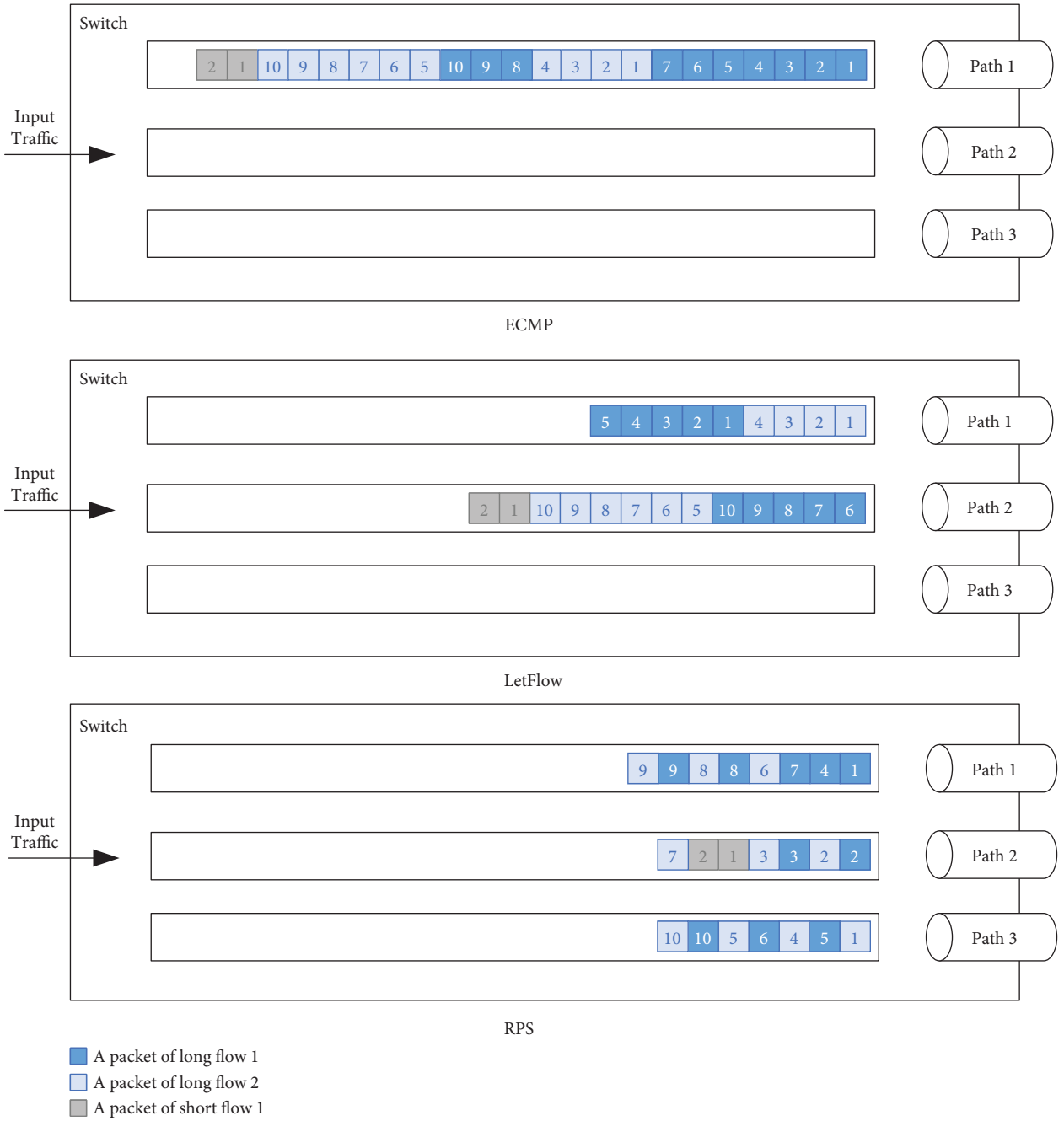

Figure 1: Flow-level transmission performances of heterogeneous data flows under ECMP, LetFlow, and RPS.

serious packet retransmissions, thus leading to significant performance degradation.

1.2. Performance Impairments. In order to further verify the above analysis, we conducted simulation experiments on NS2. Specifically, there are 40 hosts in total, and these hosts are divided into two groups (A and B). A total of 10 parallel paths are constructed through 10 spin switches, and each link has 1 Gbps bandwidth and a 25μs delay, which leads to the 200μs round-trip propagation delay (RTPD) between group A and group B. All the switch buffer sizes are set to 192 KB. The detailed network topology is shown in Figure 2.

To observe how the long flow affects the transmission performance of short flows, we let the hosts in group A send many long flows and 100 short flows to the hosts in group B and gradually change the number of long flows from 1 to 10. All the long and short flows start their transmissions by following the Poisson process. On the contrary, in the experiments about how short flows affect the performance of long flows, group A sends 10 long flows, and the number of

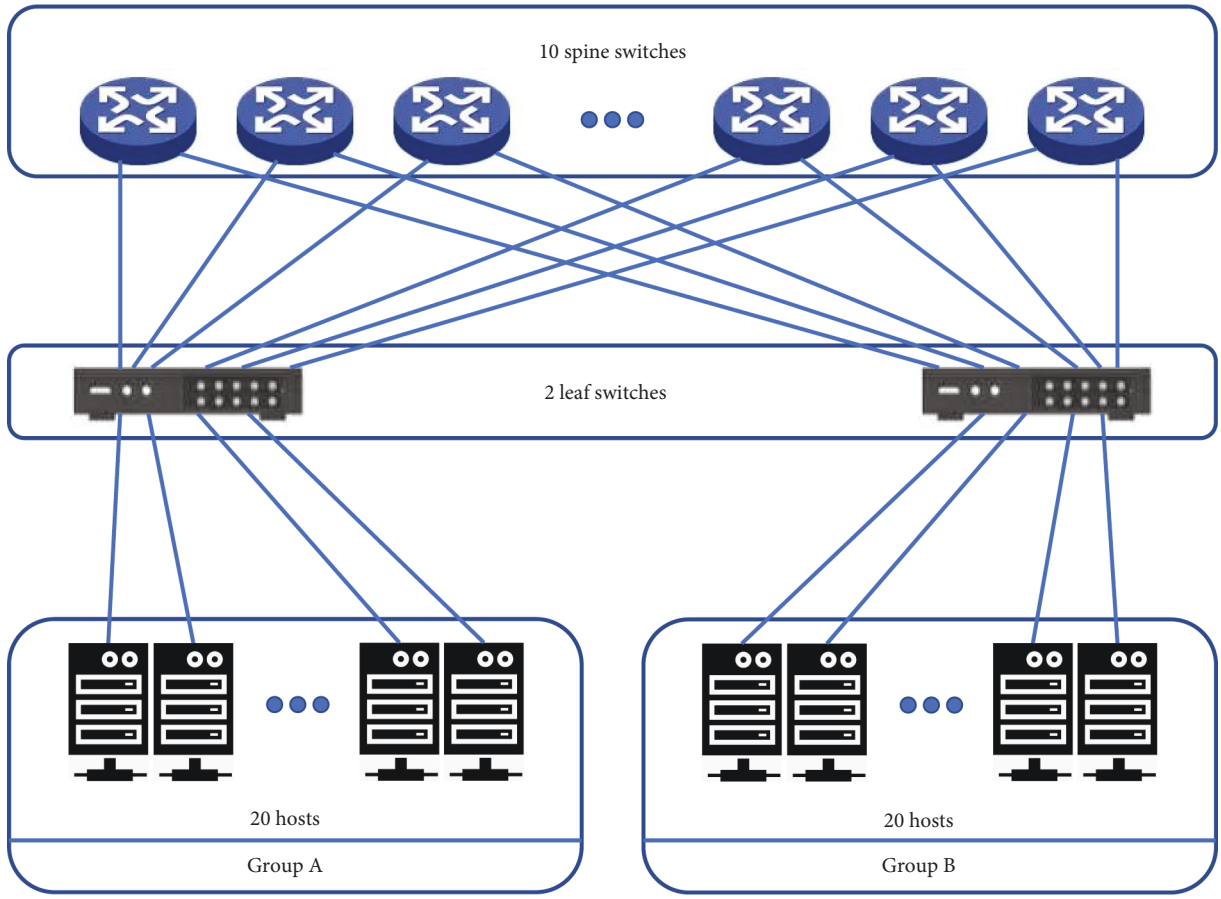

Figure 2: The leaf-spine topology.

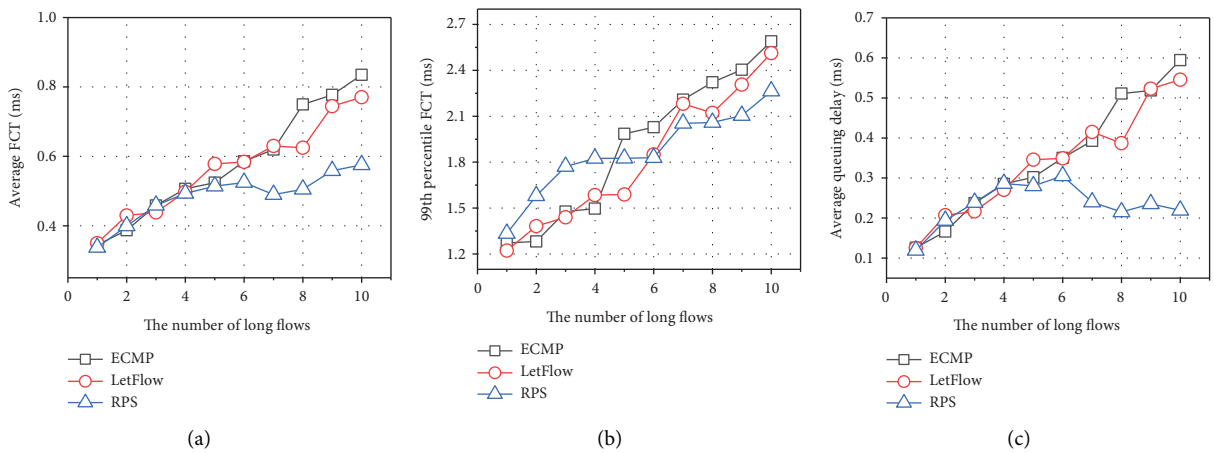

Figure 3: How long flows affect the performance of short flows? (a) How long flows affect the AFCT of short flows? (b) How long flows affect the Tail FCT of long flows? (c) How long flows affect the queuing delay experienced by short flows?

short flows is increased from 10 to 100. All the above experiments are repeated many times to obtain statistical results.

We mainly count the average flow completion time (AFCT) and the 99th percentile flow completion time (Tail FCT) for short flows, as well as the average goodput for long flows. To deeply dig into the reasons for performance degradation, we also trace some other metrics in terms of flow-level transmission performance, such as the average packet queuing delay, the number of retransmission packets, and the standard deviation of long flows' throughput on each parallel path.

Figure 3 shows the impact on various performances of short flows when the number of long flows changes. With an increasing number of long flows, the performance of short flows under ECMP and LetFlow is greatly weakened. While

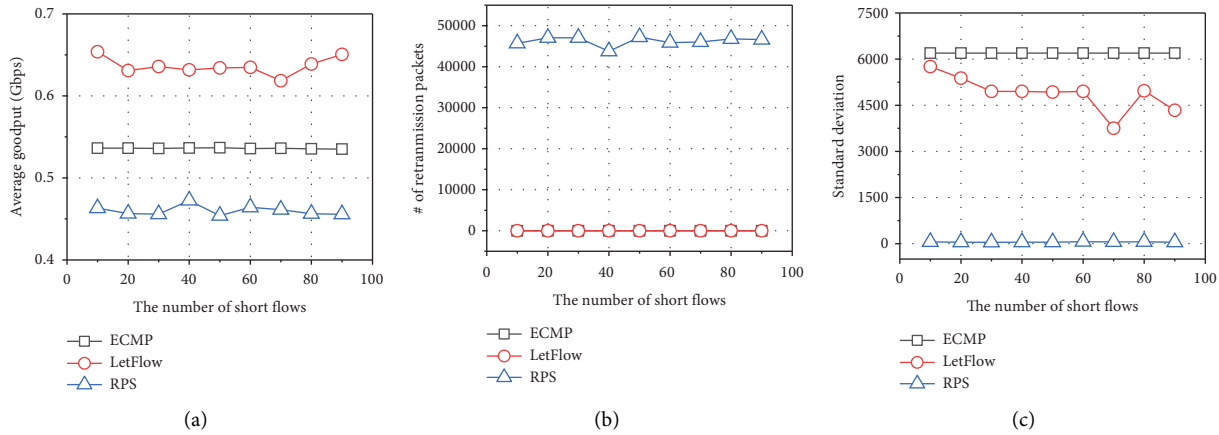

Figure 4: Effect of short flow number on long flow performance. (a) The number of short flows affects the average goodput of long flows. (b) The number of short flows affects the number of retransmission packets of long flows. (c) The number of short flows affects the standard deviation of long flows' routing.

under RPS, the performance of short flows is relatively less affected by the traffic intensity of long flows. In Figures 3(a) and 3(b), it can be observed that the AFCTs and Tail FCTs of ECMP and LetFlow increase linearly with involving more long flows in the tests. For RPS, when the number of long flows increases to a certain extent, the increase of AFCT of short flows slows down to a certain extent. This result can be explained in Figure 3(c). For ECMP and LetFlow, increasing the number of long flows leads to a rise in the average queuing delay experienced by short flows. On the contrary, the queuing delay under RPS is relatively stable, and its short flows' FCTs do not increase too much as the number of long flows increases.

Figure 4shows the impact of changing the number of short flows on the performance of long flows. Overall, the results fluctuate a little with changing the number of short flows. It can be observed from Figure 4(a) that the goodput of ECMP, LetFlow, and RPS show a gradient arrangement, and the former two coarse-grained schemes perform significantly better than the latter one. Because LetFlow has better path utilization, its goodput is also greater than that of ECMP. Figure 4(b) presents the number of retransmission packets, which can directly affect the performance of long flows. It can be noted that ECMP and LetFlow do not cause any packet retransmission. Figure 4(c) is the standard deviation of long flows' packets transmitted on each parallel path. The smaller the standard deviation, the higher the path utilization. The path utilization rate of RPS is very high, but because the packet reordering cannot be avoided, the goodputs of long flows greatly deteriorate. LetFlow improves the path utilization on the premise of ensuring that there is no disorder, thus significantly improving the goodputs of long flows. As a whole, we can find that the performance of long flows is hardly affected by the traffic intensity of short flows.

1.3. Summary. According to the above observation, we draw the following conclusions. (1) The existing representative data center load-balancing schemes do not distinguish between long flow and short flow, and the interaction between long and short flow on the bottleneck link always exists. (2) When long and short flows coexist on the same path, the increase in the number of long flows has a great impact on the performance of short flows, while the increase in the number of short flows has little impact on the performance of long flows. Therefore, our design objective is to solve the contradiction between long and short flow and strive to meet their requirements for transmission performance at the same time.

2. Design

This section introduces TAM in detail.

2.1. Insight. Based on the analysis and experimental results in the previous section, it can be concluded that the queuing delay is the main factor affecting the short flow's transmission performance, while the long flow's transmission performance mainly depends on the path utilization and the degree of packet reordering [26, 27]. When long and short flows coexist on the same path and compete for bandwidth, the performance of long flows is hardly affected by the traffic intensity of short flows. On the contrary, the traffic intensity of long flows can strongly affect the transmission performance of short flows. Generally, the greater the traffic intensity of long flows competing with short flows, the worse the performance of short flows will be. On the contrary, the better the performance of short flows will be.

In order to take into account the transmission performance of long and short flows, we designed a traffic-aware marking (TAM) rerouting mechanism, which proactively marks the packets of long flows to ensure that the transmission path always provides a very low queuing delay. When the long and short flows compete for the bandwidth of the bottleneck link, TAM further marks the long flows' packets to make them give up bandwidth to the short flows. In addition, TAM designs different routing strategies for

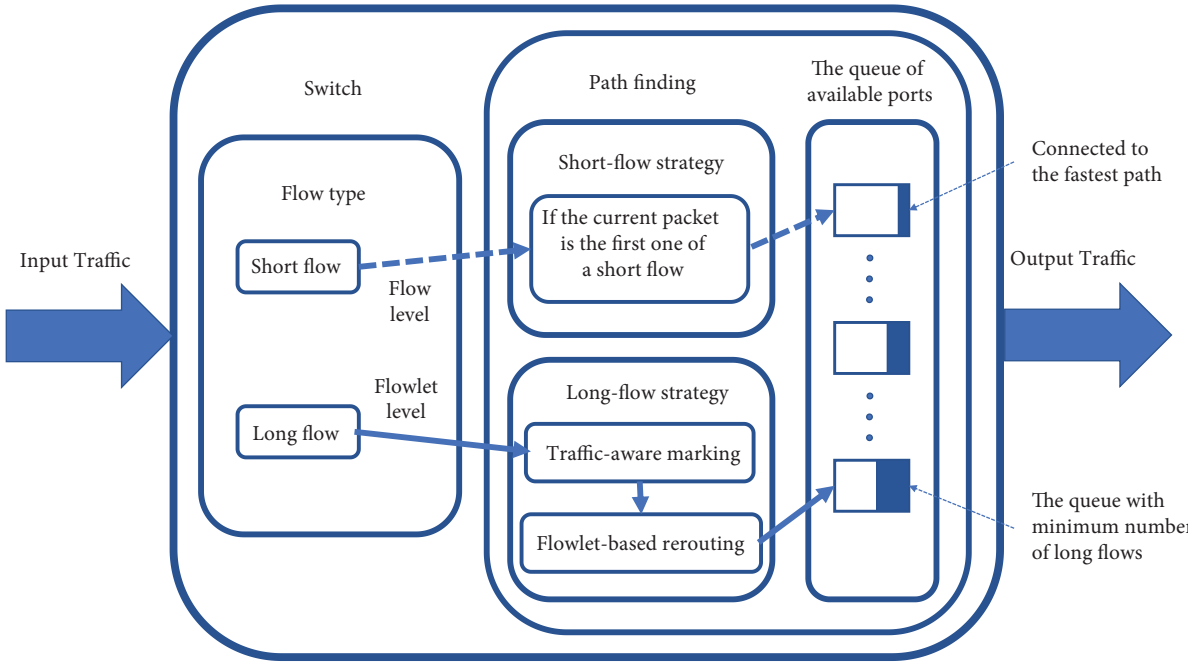

Figure 5: An overview of TAM.

long and short flows according to their different performance requirements.

TAM selects switch ports for short flows in flow units so that the short flows' packets will not be out of order during transmission. Meanwhile, the short flow selects the transmission path that can reach the destination leaf switch the fastest according to estimated transmission delays of the alternative paths. On the contrary, TAM allows the long flows to execute the rerouting mechanism in the unit of flowlet and senses whether they coexist with the short flows in the same transmission path in real time. When the short flows appear, TAM forces the long flows to retreat. When the flowlet of a long flow appears, the long flow will re-select a fast path with the least number of long flows for transmission.

2.2. Overview. TAM is only deployed on the switch without any modification to the end host. As shown in Figure 5, when a packet arrives at the switch, TAM first identifies its flow type. If the packet belongs to a short flow, TAM performs flow-level routing, which is to select the fastest available path to the destination leaf switch for transmission. If the packet comes from a long flow, TAM performs the flowlet-level rerouting and monitors the switch queue length in real time. If the queue length of the bottleneck link is greater than a predefined threshold, it immediately marks the long-flows packets arriving at the switch to keep the queue length at a very low level. In addition, when TAM detects that long and short flows coexist in the same transmission path, it immediately marks all the long flows' packets during the coexistence period, making the long flow abdicate bandwidth to the short flows. Meanwhile, when the flowlet appears, TAM reroutes the long flows' subsequent packets to those paths with light load.

2.3. Algorithm of TAM. The main operations of TAM include four parts. (1) We set the flow table to distinguish the types of flows and set the port table to record the status of flows. (2) We let the short flow choose the fastest transmission path. (3) We let the long flow perform traffic-aware marking to control the switch queue length and abdicate bandwidth to the short flow to help it complete as soon as possible when long and short flows coexist. (4) We update flow table and port table in real time.

As shown in Algorithm 1, when a data flow arrives at the switch, TAM first initializes some variables for it: sending time t_i, port number sent c_i, number of packets sent s_i, size of each packet ps_i, and queue length $k*$ ($k* = C*RTT/7$ according to the recommendation described in [2]) for sensing long flows. Next, TAM will calculate the size of the data flow and divide the data flow into long flow and short flow with 100 KB as the boundary. When the current data flow is a short flow, TAM judges whether the data packet is the first packet of the data flow. If so, TAM will select the port with the lowest transmission delay for it. If there are multiple ports with the smallest transmission delay, we select one at random. If the data packet is not the first packet of the data flow, we continue to send it according to the sending port of the previous packet. When the data flow is identified as a long flow, TAM will monitor whether the data flow is rerouted in the unit of flowlets. If the long flow is divided into multiple flowlets, the port with the least number of long flows will be selected for rerouting; if there are multiple ports with the smallest transmission delay, we select one at random. If there is no flowlet, it will continue to be sent through the port through which the previous packet passed. At the same time, TAM will also monitor the queue length of the selected port. If it exceeds the pre-set $k*$, it will mark the long flow. If the long and short flows coexist in the selected port, it

```
(1) Initialization: for the i th data flow: t_i = 0, c_i = 0, s_i = 0, ps_i, k* = C*RTT/7
(2) when switch receives a normal data packet p
(3) the data flow including p s_i++
(4)   if s_i * ps_i < 100 KB then
(5)     if p is the first packet then
(6)       do select the port with the shortest transmission delay
(7)       if port with the shortest transmission delay = 1 then
(8)         do c_i = port number
(9)       else port with the shortest transmission delay >1 then
(10)        do select the port with the least long flows for c_i
(11)        if port with the least long flows = 1 then
(12)          do c_i = port number
(13)        else port with the least long flows >1 then
(14)          do c_i = random port number
(15)    else p is not the first packet then
(16)      do follow the last packet
(17)    fi
(18) else if s_i * ps_i > 100 KB then
(19)    if flowlet exists then
(20)      do select the port with the least long flows in present
(21)      if port with the least long flows = 1 then
(22)        do c_i = port number
(23)      else port with the least long flows >1 then
(24)        do select the port with the shortest transmission delay for c_i
(25)        if port with the shortest transmission delay = 1 then
(26)          do c_i = port number
(27)        else port with the shortest transmission delay >1 then
(28)          do c_i = random port number
(29)    else p does not need to reroute then
(30)      do follow the last packet
(31)    fi
(32)    if the length of the port is selected by long flow packet p > k* then
(33)      do traffic-aware marking for long flows
(34)    else if short and long flow coexist in the port selected by p then
(35)      do traffic-aware marking for long flows
(36)    else
(37)      do no operation
(38)    fi
(39)  fi
(40) send p from port c_i
```

Algorithm 1: Pseudocode of TAM.

will also mark the long flow. Except for these two cases, TAM will not mark any long flows. Finally, the packet will be forwarded from the selected port.

2.4. Details

2.4.1. Implementation of Traffic-Aware Marking. TAM employs the ECN (Explicit Congestion Notification) [28–31] mechanism, which is currently supported by data center switches to mark packets. The specific marking process is as follows. (1) When the long flows' packets arrive at the switch, if the queue length on the using output port is greater than a predefined threshold or if the long flows coexist with short flows in the same path, TAM modifies the ECN field of the IP header of the long-flow packet to 11. (2) When the receiver receives the long-flow packet, it checks the packet's IP header. If the ECN field is 11, the receiver sets the ECE bit in the TCP header of the ACK packet to 1. (3) After the sender receives the ACK packet, it reduces the sending rate of the long flow according to DCTCP once the ECE bit of the ACK packet is 1. Note that the packets of short flows are still marked according to the default queue length threshold of DCTCP, although the marking operation in TAM is only executed for long flows. The detailed implementation of traffic-aware marking is listed in Table 1.

Note that although both TAM and DCTCP use ECN, they are quite different when looking closely. The main differences are as follows. (1) DCTCP is unaware of the traffic type. TAM identifies the different types of heterogeneous flows when a packet arrives at the switch. (2) DCTCP uses ECN to control the switch queue length, while TAM employs ECN to limit the bandwidth by long flows. Under TAM, when the short flows appear, TAM forces the

long flows to retreat. (3) DCTCP marks all arriving packets (whether they belong to short flows or long flows) when the switch queue length is greater than a predefined threshold. TAM marks all the long flows' packets during the coexistence period, which ensures the bottleneck link always provides a very low queuing delay. (4) DCTCP makes some modifications to end hosts. TAM is only deployed on the switch without any modification to end hosts. (5) Compared to DCTCP, TAM is better at providing low queuing delays for short flows.

2.5. Traffic Type Awareness. Generally, it is impossible to know in advance how much data a flow need to transmit before it completes transmission. Therefore, TAM identifies the type of data flow by counting the amount of data sent so far. When the amount of data sent by a flow exceeds 100 KB, the data flow is regarded as a long flow. Otherwise, it is a short flow. The problem with this method is that when the data flow is a long flow, it will be wrongly regarded as a short flow during transmitting its first 100 KB of data, resulting in the undesired routing operation in this period. However, because the number of long flows is usually small, this misjudgment does not often occur. In addition, the survival time of short flow is very short, so the impact is very small.

2.6. Routing Mechanism of Short Flow. The routing mechanism of short flow under TAM is as follows:

(1) In order to optimize the short-flow performance to the greatest extent and reduce the queuing delay during transmission, the path with the lowest transmission delay is selected.

(2) When the transmission delays of multiple ports are all the lowest, the one with the least number of long flows is selected from these ports for transmission.

(3) When more than one port meets the above two conditions at the same time, let the short flow randomly select one of these ports for transmission. This is to avoid a herd effect; that is, multiple short flows entering the same port, resulting in the rapid growth of the queue in a short time.

(4) In order to select the port with the lowest transmission delay, TAM measures the transmission delay of the path corresponding to each port. The transmission delay is the sum of the port queuing delay and the path propagation delay.

Note that TAM estimates the path transmission delay based on the local queuing delay at the leaf switch and the propagation transmission delay between the leaf switches. The former delay can be directly measured in real time according to the accessible local status, while the latter one is generally constant and easy to measure when the network is lightly loaded. On the contrary, under TAM, the queuing delay at the spine switch is very small and can be ignored when estimating the path transmission delay since TAM maintains the queue length at the spine switch at a very low level.

2.7. Routing Mechanism of Long Flow. To make up for the performance loss caused by the active marking of long flows as much as possible, TAM employs flowlet-based rerouting to make the long flow traffic evenly distributed in all available paths during transmission. The details are as follows:

(1) When a long flow's flowlet appears, TAM chooses the port with the least total number of long flows for rerouting

(2) When there are multiple ports satisfying condition 1, the port with the lowest queuing delay is selected for rerouting

(3) When multiple ports meet conditions 1 and 2 at the same time, TAM allows the long flow to randomly select one of these ports for transmission to avoid a herding effect

2.8. Maintenance of Data Flow Status and Port Status. The TAM allows the switch to maintain a flow table and a port table to record the status of data flow and switch port, respectively.

The specific information recorded in the flow table (see Table 2) is as follows:

(1) Forwarding time: we record the time when the latest packet of each flow arrives at the switch.

(2) Forwarding port number: we record the port number by using the latest packet in current flow.

(3) Number of forwarded packets: we record the number of packets that have been sent by the current flow.

(4) Activity sign: when the current flow is a long flow, this entry records whether it is active in the current RTT. If it is, this entry is set to 1; otherwise, it is set to 0.

(5) Statistical sign: when the current flow is a long flow, this entry records whether it has been counted for a long time. If it has, the entry is set to 1; otherwise, the entry is set to 0.

The specific functions of the port information recorded in the port table (see Table 3) are as follows:

(1) The time of updating port status in previous period

(2) The number of active long flows transmitting in current port

(3) The total number of long flows currently being transmitted by the port

(4) Arrival time of the last short flow packet of the port

(5) Whether there is short flow transmission on the current port

2.9. Effectiveness Analysis. TAM divides the data flows into short and long flows and makes them adopt a coarse-grained rerouting strategy to minimize the impact of packet reordering on the transmission performance. In addition, TAM adopts different rerouting strategies for

Table 1: Implementation of traffic-aware.

Status	Traffic aware marking	
	Long flow	Short flow
Only long flows exist and switch queue length < threshold	Dont mark	Dont mark
Only short flows exist and switch queue length < threshold	Dont mark	Dont mark
Short flows and long flows coexist and switch queue length < threshold	Mark	Dont mark
Only long flows exist and switch queue length ≥ threshold	Mark	Dont mark
Only short flows exist and switch queue length ≥ threshold	Dont mark	Mark
Short flows and long flows coexist and switch queue length ≥ threshold	Mark	Mark

Table 2: The content of the flow table.

Flow ID	Forwarding time	Forwarding port number	Number of forwarded packets	Activity sign	Statistical sign

Table 3: The content of the port table.

Port ID	The time of updating port status in previous period	The number of active long flows transmitting in current port	The total number of long flows currently being transmitted by the port	Arrival time of the last short flow packet into the port	Whether there is short-flow transmission on the current port

the different performance requirements of the two heterogeneous flows: short flows are allowed to choose the transmission path with the least delay and the last long flows when they arrive at the source leaf switch. This design can ensure that the short flows avoid the impact of competing with long flows on their performance, thus greatly reducing their flow completion time. On the contrary, TAM also allows the long flows to carry out the traffic-aware marking to keep the switch queue length at a very low level, thus further reducing the short flows' FCT. In addition, TAM employs the flowlet-level rerouting granularity to eliminate packet reordering and improves the path utilization of long flows to ensure that their performance is not adversely affected by traffic-aware marking.

3. Deployment

The control logic of TAM mainly involves traffic-aware path selection, flow and port status maintenance, while the existing Commercial Off The Shelf (COTS) switch is difficult to meet the requirements since its reconfigurability is poor and its on-chip memory is limited [9, 10, 32]. Fortunately, the programmable switch, represented by the Tofino switch, which can implement complex processing logic, is becoming more commonplace nowadays. Correspondingly, TAM can be deployed on the programmable switch, which is built on the P4-programmable Protocol Independent Switch Architecture (PISA) [33]. Instead of using traditional physical methods to bake logic units into silicon, the new logic resides in the P4 program [33], thus reducing the difficulty of adding TAM logic to the switch by the user or manufacturer. Moreover, many sketch-based methods can reduce the cost of network measurement tasks [33, 34]. TAM can combine these advantages to further reduce its deployment overhead.

4. Evaluation

In this section, we evaluate the performance of TAM based on numerous NS2 simulation tests. Firstly, we redo the test in Section "Motivation" by introducing two other representative data center load-balancing mechanisms (DRB and Presto) to observe whether TAM can improve the performance of long and short flows as expected. Next, we evaluate the performance of TAM in asymmetric scenes. Then, we conducted large-scale simulation experiments based on a variety of real data center loads to evaluate the comprehensive performance of TAM in real application scenarios.

We briefly introduce the mechanisms evaluated in this section as follows:

ECMP routes the data flow according to the hash of the original IP address.

LetFlow pre-sets an interval threshold. When the switch senses that the interarrival time of two packets belonging tothe same flow is greater than the threshold, the subsequent packets of the data flow will be migrated to another path.

RPS randomly sends all data flows to all parallel paths in packets.

DRB sends all data flows to all parallel paths in packets by polling.

Presto divides the data flow into flow cells into 64 KB units and then sends them in the form of polling.

4.1. Impairment Microbenchmarks. This section mainly shows the performance improvement of TAM by redoing the tests in Section 1.2.

Figure 6(a) shows the average completion time of short flow. As a coarse-grained mechanism, Presto has little difference from EMP and LetFlow, and its overall performance is between ECMP and LetFlow. Meanwhile, the performance

of short flow in DRB is not ideal. The average completion time of short flows in TAM is significantly lower than those of the other protocols, and it is not affected by the number of long flows, which is particularly prominent in Figure 6(b). The 99th percentile completion time of TAM is hardly affected by the number of long flows, indicating that TAM reduces the impact of long flows on the performance of short flows to a very considerable extent. It can be observed from Figure 6(c) that the average queue delay of short flows in TAM is basically at the same level as that of RPS, which effectively reduces the queuing delay that short flows need to experience.

Figure 7 shows how the traffic intensity of short flows affects the performance of long flows. The results are consistent with the conclusions of the analysis in the motivation section. As shown in Figure 7(a), increasing the number of short flows is difficult to affect the performance of long flows. Although Presto's flowcell is 64 KB, blind rerouting will still lead to serious packet reordering, thus reducing the goodput of long flows. The goodputs of long flows in Presto and DRB are lower than those of RPS. The goodput of long flows in TAM is not only greater than LetFlow but also very stable. It can be observed from Figure 7(b) that the number retransmission packets of TAM, ECMP, and LetFlow are all 0 and not affected by the number of short flows again, showing the superiority of coarse-grained mechanism in avoiding packet reordering. In Figure 7(c), we can see that TAM also has higher path utilization than LetFlow and ECMP. Coupling with effectively avoiding packet reordering, as shown in Figure 7(b), the average goodput of long flows in TAM is the best.

4.2. Adaptability in an Asymmetrical Scenario. Asymmetry is common in production data centers, which generally operate under rich uncertainties, such as link failures and heterogeneity in-network equipment. Adapting to asymmetry gracefully, therefore, is important. This raises the question: is TAM resilient to asymmetry? In this section, we evaluate the adaptability of TAM in asymmetric scenes. We followed the same topology as the symmetric scenario as shown in Figure 1, but divided all parallel paths into good and bad paths. The link delay of the good path is 25μs while the link delay of the bad path is 4 times that of the good path (100μs). By observing Figures 8(a) and 8(b), we can find that, under TAM, the AFCTs and 99th percentile FCTs of short flows are lower than the other mechanisms. The reason is that TAM can perceive the path state, while the other mechanisms do not distinguish between good and bad paths. Figure 8(c) presents the stably low queuing delay experienced by short flows under TAM, which again guarantees their performance requirements in asymmetric scenarios.

The results in Figure 9 show that the performance of long flows is still not easily affected by increasing the number of short flows. Compared with the results in Section 1.2, we can find that the polling transmission method is better than random fashion. Presto's average goodput of long flows not only exceeds ECMP but also is even almost equal to LetFlow. And the long-term performance of DRB also exceeds that of RPS. As a whole, TAM still greatly outperforms the other representative data center load balancing mechanisms.

4.3. Large-Scale Simulation

4.3.1. Performance in Real Data Center Workload. In this section, we will build a large-scale leaf-spine network that includes 10 leaf switches and 10 spine switches. Each link has a 40 Gbps [35, 36] bandwidth and a 1μs delay. The switch buffer is set to 375 KB. Therefore, the round-trip propagation delay between each pair of hosts is 8μs. There are 200 hosts in total; we randomly choose half of them as senders to send DCTCP flows to the rest of the hosts. For a comprehensive performance comparison, we set up 5 real data center workloads, such as Cache Follower [37], Web server [38, 39], Web Search [40–42], data mining [43–45], and Hadoop [46, 47]. Their respective distributions of flow data sizes are shown in Table 4. In each test, we randomly send a total of 10,000 data flows by setting their starting time according to Poisson process.

The results of short flow performance can be observed in Figures 10(a) and 10(b). AFCTs and Tail FCTs under Cache Follower are significantly higher than other types of workloads [48, 49]. From Table 4, we can see that the proportion of long flows in the Cache Follower is the highest, so when the number of data flows is the same, its load of the heaviest in the five real workloads. For this reason, the queuing delay experienced by short flows may be greater. However, both RPS and DRB are fine-grained mechanisms, and the impact of workload on queue length is not obvious. Therefore, the short-term performance of these two mechanisms is much higher than those of ECMP, LetFlow, and Presto. The reason for Web Server and Hadoop is that there is no extreme long flow (>1 MB) or the proportion of extreme long flow is very low, which leads to the low queue length on the bottleneck link, so that the result difference of each load balancing mechanism is not obvious.

Figures 11(a) and 11(b) present the large-scale test results of long flow performance. The differences under five different workloads are also concentrated on Cache Follower and Web Server workloads. There is no extreme long flow in the Web Server workload, so the performance of its long flow is the best due to the relatively lowest load. The Cache Follower has the heaviest load and the proportion of long flows to be transmitted is too high. Therefore, the performance is affected by the long flow itself.

In general, compared with coarse-grained schemes (ECMP, LetFlow, and Presto), RPS and DRB can better ensure the performance requirements of short flow because fine-grained schemes make the load of parallel paths more balanced, the path utilization higher, and the queuing delay of switches lower. On the contrary, the coarse-grained scheme ensures the performance requirements of long flow because it rarely causes packet reordering.

In workloads with a low proportion of long flows, such as Web Server and Hadoop, the performances of short flows are similar to those of the other load balancing schemes due to light load, but the traffic-aware marking strategy not only

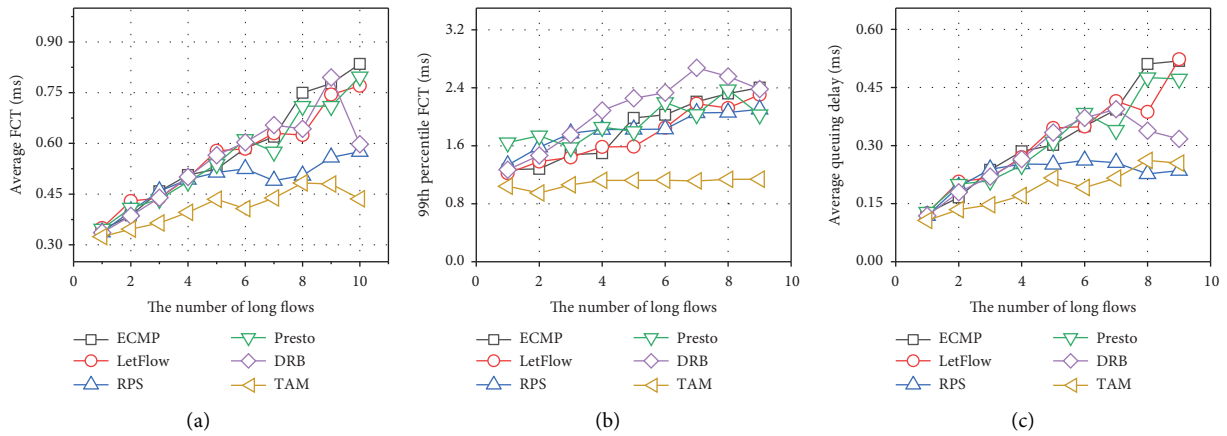

Figure 6: Comparison of the long flow number affecting short flow between TAM and 5 typical load balancing mechanisms in a symmetrical scenario. (a) The number of long flows affects the average FCT of short flows. (b) The number of long flows affects the 99th percentile FCT of short flows. (c) The number of long flows affects the average queuing delay of short flows.

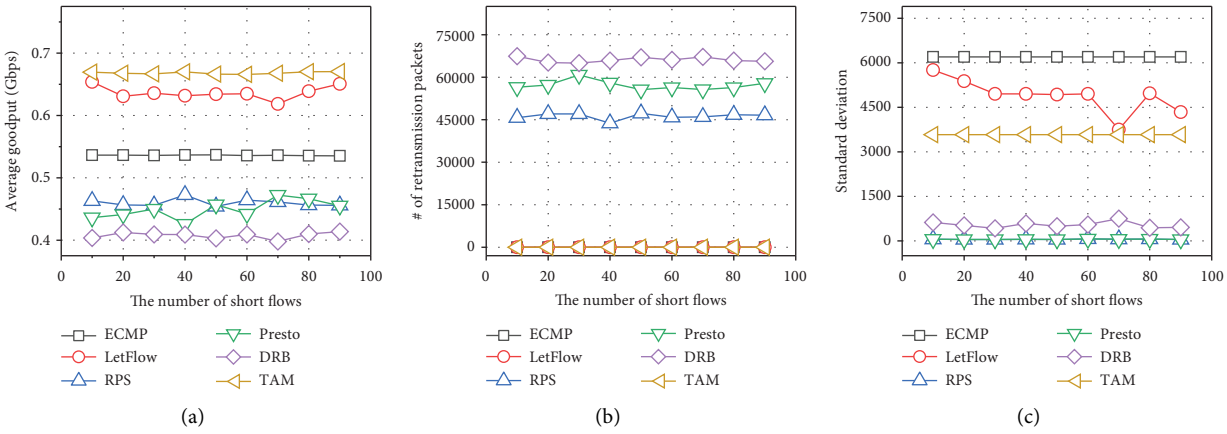

Figure 7: Comparison of short flow numbers affecting long flow between TAM and 5 typical load balancing mechanisms in a symmetrical scenario. (a) The number of short flows affects the average goodput of long flows. (b) The number of short flows affects the number of retransmission packets of long flows. (c) The number of short flows affects the standard deviation of long flows' routing.

ensures that short flows experience low queuing delays but also achieves a good balance in avoiding packet reordering and improving path utilization with the help of TAM's routing strategy. Therefore, the performance of TAM is better than these representative load-balancing schemes.

4.4. Performance in Mixed Data Center Workloads. In order to further evaluate the performance of TAM, we mixed 10000 of the five real loads of Table 1, a total of 50000 DCTCP data flows, using the same topology as in the previous section. All flows start by following a Poisson process, and we divide the load intensity from 0.2 to 0.8 in steps of 0.2 according to the method described in [47].

Figures 12(a) and 12(b) present the performance of short flows. We find when the load intensity is low, the performance of short flows under TAM is near those under fine-grained schemes (RPS and DRB). With the load intensity going heavy, except for TAM, the performance of short flows under 5 representative load balancing schemes prominently worsens. In contrast, TAM always keeps the AFCTs and tail FCTs of short flow at a low level, which does not change evidently with the load intensity. In Figures 12(c) and 12(d), the performance of long flows under TAM is comprehensively superior to the other schemes, although it has also decreased significantly with the increase of load intensity. Therefore, combining the performance of long and short flows of TAM and according to the results in Figure 12(e), we can find that the overall performance of all data flows in TAM is better than the other schemes. Although the decline of data flows' performance slows down in each scheme with the increase of load intensity, the overall performance of TAM is improved by more than 20% compared with other schemes.

5. Related Work

In recent years, researchers have proposed many transport protocols to improve transmission performance. However, due to the inability to make full use of the bandwidth resources of the data center, the network performance cannot meet the expectations. Therefore, various load-balancing

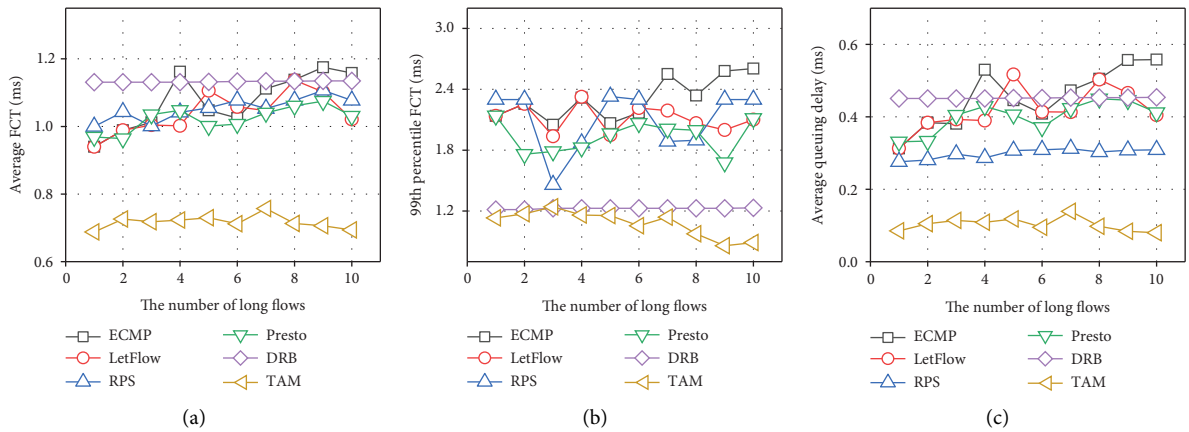

Figure 8: A comparison of the long flow number affecting short flow between TAM and 5 typical load balancing mechanisms in an asymmetrical scenario. (a) The number of long flows affects the average FCT of short flows. (b) The number of long flows affects the 99th percentile FCT of short flows. (c) The number of long flows affects the average queuing delay of short flows.

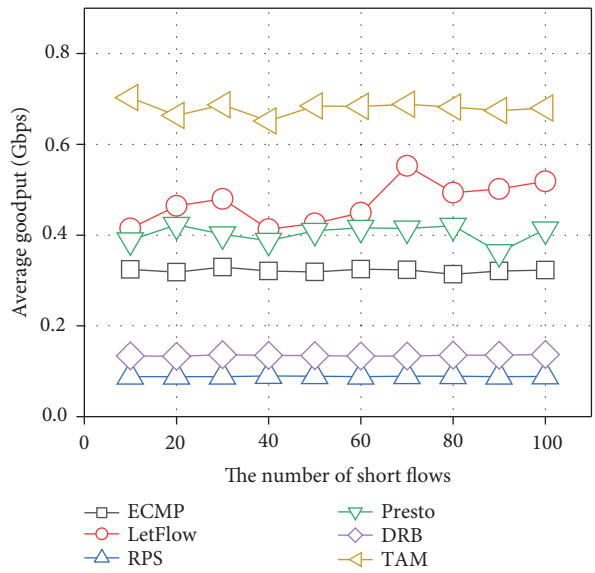

Figure 9: A comparison of the short flow number affecting long flow between TAM and 5 typical load balancing mechanisms in an asymmetrical scenario.

mechanisms based on data center networks are proposed to optimize multipath parallel data transmission and realize low completion time and high throughput of data flow by providing greater bisection bandwidth.

Flow-level load-balancing schemes: ECMP [17] is the de-facto flow-level load-balancing scheme. By hashing the five tuples in the packet header, the packet is mapped to one of multiple equivalent parallel links. However, ECMP has the well-known problem of congestion caused by hash collisions and cannot deal with congestion. Therefore, the path utilization is not enough, resulting in the performance of short flows cannot be met. To avoid hash collision problems, many other flow-lever load-balancing schemes are proposed. Hedera [50] uses a central controller to adaptively schedule flows across multistage switching fabrics to efficiently utilize available link capacity. MicroTE [51] leverages the short-term and partial predictability of the traffic matrix to dynamically schedule flows in the OpenFlow fabric to avoid hotspots. FlowBender [52] detects congestion at the end hosts and then reroutes the specific flows to react to congestion and link failures.

Packet-level load-balancing schemes: RPS [18] and DRB [19] are two representative packet-level load balancing mechanisms. RPS randomly routes each packet arriving at the switch, while DRB sends all packets by polling. CAPS [53] is a coding-based adaptive packet spraying scheme, that efficiently addresses the packet reordering problem by using forward error correction technology. TLB [54] is a traffic-aware adaptive granularity load-balancing scheme, which dynamically adjusts the switching granularity of long flows to meet the deadline of short flows. To adapt to the various path diversities, AG [55] adjusts rerouting granularity based on the asymmetric degree of multiple parallel paths. Hermes [56] uses comprehensive sensing to detect path conditions and uncertainties and then timely and cautiously makes load balancing decisions only when it will be beneficial.

Flowlet-level load-balancing schemes: LetFlow [20] is a very typical flowlet-level load-balancing mechanism. Its basic idea is to set a preset threshold to reflect the congestion degree of the transmission path. When the transmission interval between two consecutive packets in the same data flow exceeds this threshold, it is considered that the current transmission path has begun to be congested and the remaining traffic needs to be rerouted to other paths. LetFlow routing and rerouting are determined by random selection, so this mechanism does not completely solve the problem of insufficient path utilization because it does not perceive the path status. CONGA [15] also performs rerouting at the flowlet level by using in-network link load information to avoid packet reordering and improve link utilization. Flowtune [57] uses a centralized controller to assign the optimal transmission rate for each flowlet. Clove [58] uses a weighted round-robin manner to forward each

Table 4: Respective distribution of flow data sizes.

Flow type	Data mining (%)	Web search (%)	Cache follower (%)	Web server (%)	Hadoop
0–10 KB	78	59	50	68	94
10–100 KB	5	3	3	18	1
100KB-1 MB	8	18	18	14	2
>1 MB	9	20	29	–	3

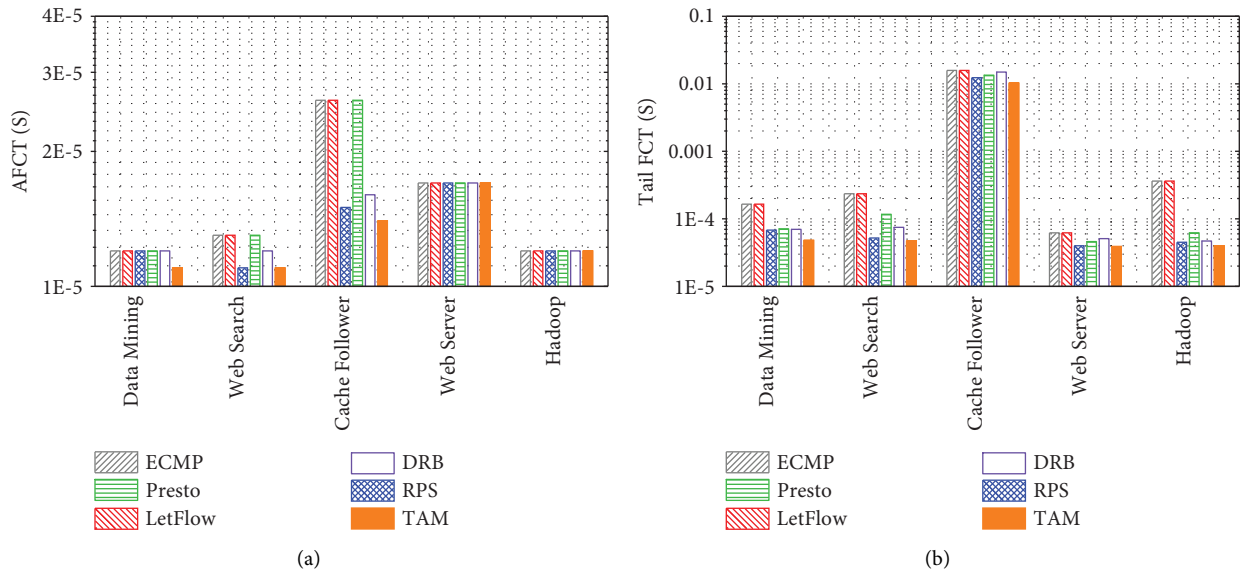

Figure 10: Short flows' performance in a large-scale scenario. (a) AFCTs of short flows in 5 workloads. (b) Tail FCTs of short flows in 5 workloads.

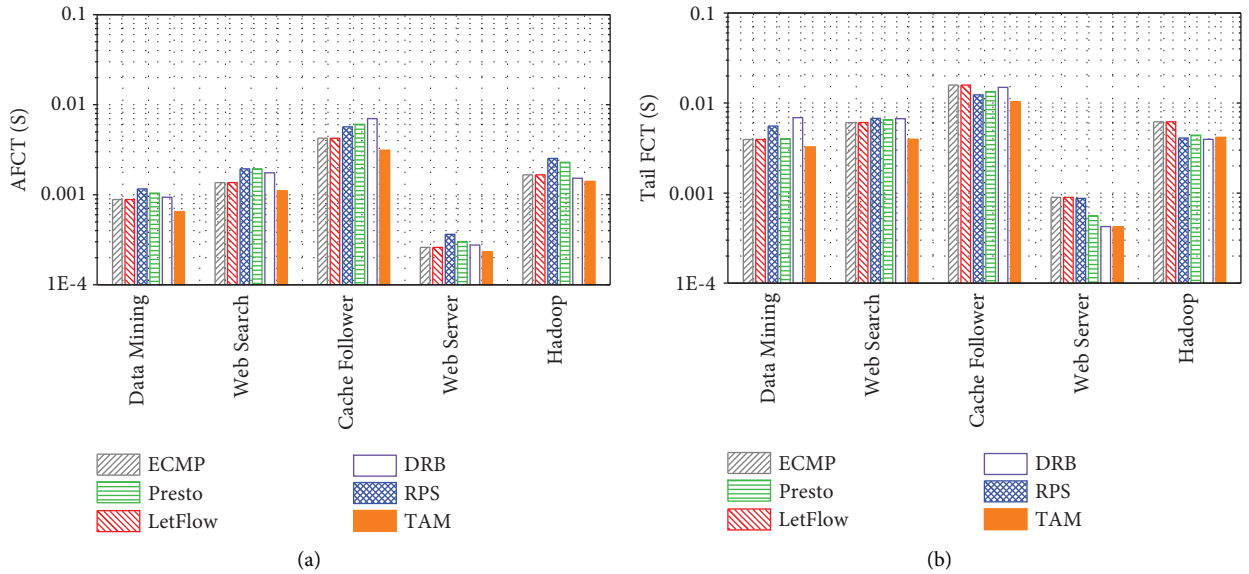

Figure 11: Long flows' performance in a large-scale scenario. (a) AFCTs of long flows in 5 workloads. (b) Tail FCTs of long flows in 5 workloads.

flowlet to the end hosts. Expeditus [59] detects local congestion and designs a two-stage path selection scheme to select optimal forwarding paths. HULA [60] forwards flowlets to the best path based on the hop-by-hop congestion state at the data plane of the programmable switch.

Flowcell-level load-balancing schemes: Presto [21], as a per-flowcell load-balancing scheme, adopts the same polling

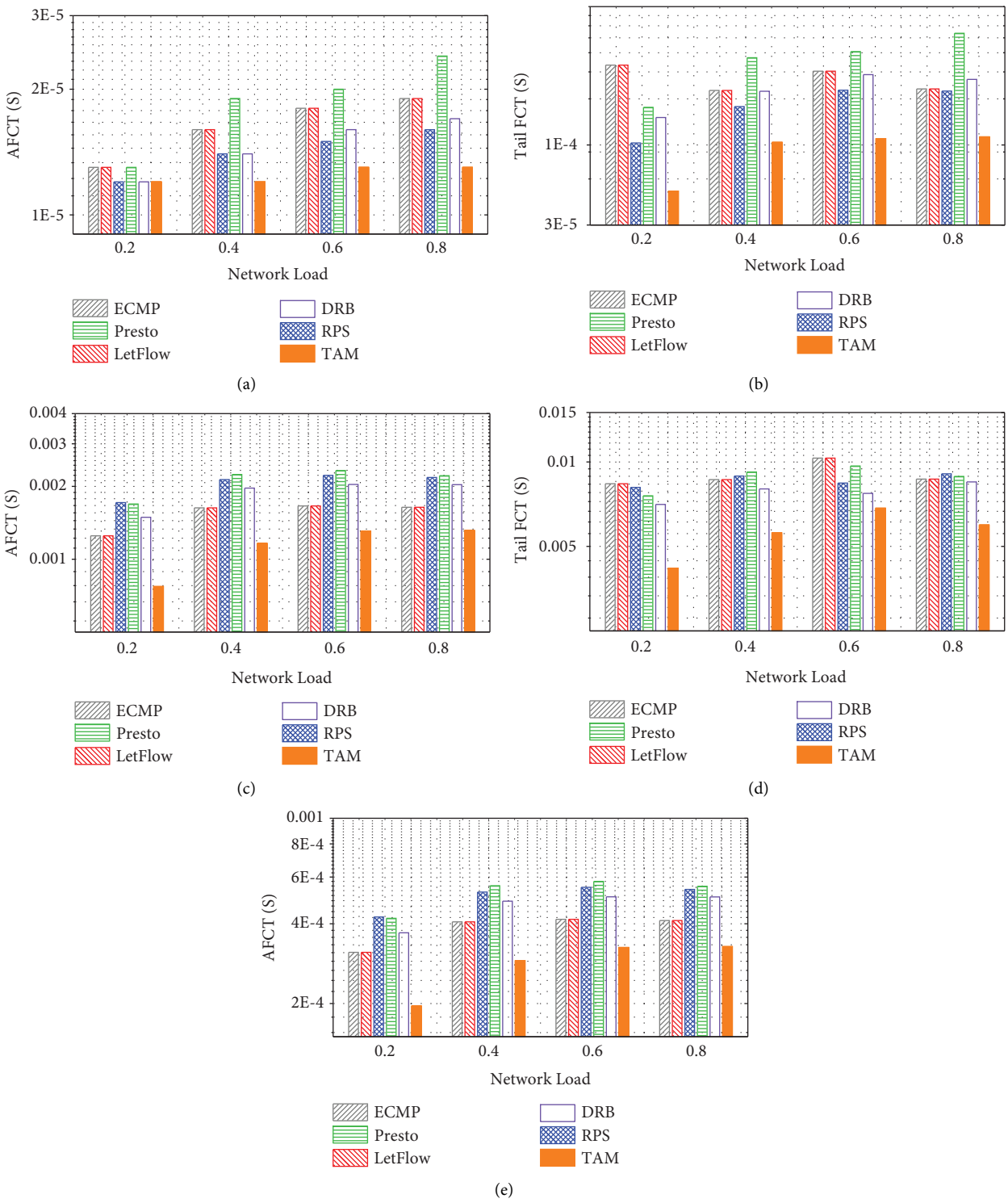

Figure 12: The performance of data flows in a mixed scenario. (a) AFCTs of short flows. (b) Tail FCTs of short flows. (c) AFCTs of long flows. (d) Tail FCTs of long flows. (e) Overall performance of data flows.

sending method as DRB. The difference is that Presto sends data flows with 64 KB as a flowcell. Due to the blindness of this traffic segmentation and rerouting method, Presto's performance is very unstable.

Different from the above load-balancing mechanisms, our proposed TAM improves the short flow performance by actively controlling the queue length by judging whether the long and short flows coexist on the same path and whether the queue length exceeds the pre-set value. At the same time, we monitor the status of each path and select the path with lower traffic for the data flow that meets the traffic conditions of LetFlow segmentation to

improve the throughput of long flow and avoid packet reordering at the same time.

6. Conclusion

This work designs a new load-balancing scheme, TAM, to meet the performance requirements of both long and short flows at the same time. TAM marks the long flow in the path where the long and short flow coexist and the queue length exceeds the threshold. Meanwhile, the status of each parallel path is monitored to select a better transmission path for flows or flowlets. Through a series of large-scale NS2 simulation tests, the results show that TAM can improve the long and short flow's performance by 20%–60% compared with the state-of-the-art data center load balancing schemes.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Ran Huang and Jingliang Zhang are cofirst authors of this article.

Acknowledgments

The authors would like to thank Dr. Yijun Li for his valuable help in the evaluation. This work was supported by the National Natural Science Foundation of China, under Grants 61872403, 62102047, and 62102046, and in part by the Hunan Province Key Laboratory of Industrial Internet Technology and Security, under Grant 2019TP1011, and the Research Foundation of Education Bureau of Hunan Province, China (20C0030).

References

[1] Y. Jiang, L. Sivalingam, S. Nath, and R. Govindan, "Webperf: evaluating what-if scenarios for cloud-hosted web applications," in *Proceedings of the. ACM SIGCOMM*, pp. 258–271, New York, NY, USA, August 2016.

[2] M. Alizadeh, A. Greenberg, D. A. Maltz et al., "Data center TCP (DCTCP)," *ACM SIGCOMM - Computer Communication Review*, vol. 40, no. 4, pp. 63–74, 2010.

[3] J. Huang, Y. Huang, J. Wang, and T. He, "Adjusting packet size to mitigate TCP incast in data center networks with COTS switches," *IEEE Transactions on Cloud Computing*, vol. 8, no. 3, pp. 749–763, 2020.

[4] H. Zhang, X. Shiy, X. Yin, F. Ren, and Z. Wang, "More load, more differentiation-a design principle for deadline-aware congestion control," in *Proceedings of the. IEEE INFOCOM*, pp. 127–135, Hong Kong, China, April 2015.

[5] Y. Wang, K. Springborn, C. Alfeld, M. Ryan, D. Wetherall, and A. Vahdat, "Swift: delay is simple and effective for congestion control in the datacenter," in *Proceedings of the ACM SIGCOMM*, pp. 514–528, New York, NY, USA, July 2020.

[6] H. Xu and B. Li, "RepFlow: minimizing flow completion times with replicated flows in data centers," in *Proceedings of the. IEEE INFOCOM*, pp. 1581–1589, Toronto, ON, Canada, April 2014.

[7] C. Lee, C. Park, K. Jang, S. Moon, and D. Han, "DX: latency-based congestion control for datacenters," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 335–348, 2017.

[8] T. Zhang, J. Wang, J. Huang, J. Chen, Y. Pan, and G. Min, "Tuning the aggressive TCP behavior for highly concurrent HTTP connections in intra-datacenter," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3808–3822, 2017.

[9] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "Practical information-agnostic flow scheduling in data center networks," in *Proceedings of the USENIX NSDI*, pp. 455–468, Oakland, CA, USA, May 2015.

[10] A. Sivaraman, C. Kim, R. Krishnamoorthy, and A. Dixit, "DC.p4: programming the forwarding plane of a data-center switch," in *Proceedings of the ACM SOSR*, New York, NY, USA, June 2015.

[11] J. Zhang, "Deadline-aware bandwidth sharing by allocating switch buffer in data center networks," in *Proceedings of the. IEEE INFOCOM*, pp. 1–9, Toronto, ON, Canada, April 2016.

[12] Z. Zhao, X. Shi, X. Yin, Z. Wang, and Q. Li, "HashFlow for better flow record collection," in *Proceedings of the. IEEE ICDCS*, pp. 1–12, Dallas, TX, USA, July 2019.

[13] T. Zhang, J. Wang, J. Huang, Y. Huang, J. Chen, and Y. Pan, "AdaptiveAcceleration data center TCP," *IEEE Transactions on Computers*, vol. 64, no. 6, pp. 1–1533, 2014.

[14] S. Zou, J. Huang, J. Wang, and T. He, "Flow-aware adaptive pacing to mitigate TCP incast in data center networks," *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 1–14, 2020.

[15] M. Alizadeh, T. Edsall, S. Dharmapurikar et al., "CONGA: distributed congestion-aware load balancing for datacenters," *ACM SIGCOMM - Computer Communication Review*, vol. 44, no. 4, pp. 503–514, 2015.

[16] Z. Guo, S. Liu, and Z. L. Zhang, "Traffic control for RDMA-enabled data center networks: a survey," *IEEE Systems Journal*, vol. 14, 2019.

[17] C. E. Hopps, "Analysis of an equal-cost multi-path algorithm," in *RFC 2992*, pp. 2006-2007, 2000.

[18] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the impact of packet spraying in data center networks," in *Proceedings of the. IEEE INFOCOM*, pp. 2130–2138, Turin, Italy, April 2013.

[19] J. Cao, R. Xia, P. Yang et al., "Per-packet load-balanced, low-latency routing for clos-based data center networks," in *Proceedings of the. ACM CoNEXT*, pp. 49–60, New York, NY, USA, December 2013.

[20] E. Vanini, R. Pan, M. Alizadeh, P. Taheri, and T. Edsall, "Let it Flow: Resilient Asymmetric Load Balancing with Flowlet Switching," in *Proceedings of the. USENIX NSDI*, pp. 407–420, Boston, MA, USA, March 2017.

[21] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella, "Presto: edge-based load balancing for fast data-center networks," *ACM SIGCOMM - Computer Communication Review*, vol. 45, no. 4, pp. 465–478, 2015.

[22] T. Benson, A. Akella, and D. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the. ACM IMC*, pp. 267–280, New York, NY, USA, November 2010.

[23] L. I. Chen, K. Chen, W. Bai, and M. Alizadeh, "Scheduling mix-flows in commodity datacenters with karuna," in *Proceedings of the ACM SIGCOMM*, pp. 174–187, New York, NY, USA, August 2016.

[24] W. Bai, S. Hu, K. Chen, K. Tan, and Y. Xiong, "One more config is enough: saving (DC)TCP for high-speed extremely shallow-buffered datacenters," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 489–502, 2021.

[25] W. Bai, K. Chen, L. Chen, C. Kim, and H. Wu, "Enabling ECN over generic packet scheduling," in *Proceedings of the. ACM CoNEXT*, pp. 191–204, New York, NY, USA, December 2016.

[26] T. Zhang, Q. Zhang, Y. Lei, S. Zou, J. Huang, and F. Li, "Load balancing with traffic isolation in data center networks," *Future Generation Computer Systems*, vol. 127, pp. 126–141, 2022.

[27] T. Zhang, R. Huang, Y. Hu et al., "Load balancing with deadline-driven parallel data transmission in data center networks," *IEEE Internet of Things Journal (Early Access)*, p. 1, 2022.

[28] G. Zeng, W. Bai, G. Chen et al., "Congestion control for cross-datacenter networks," in *Proceedings of the. IEEE ICNP*, pp. 1–12, Chicago, IL, USA, October 2019.

[29] J. Zhang, W. Bai, and K. Chen, "Enabling ECN for Datacenter Networks with RTT Variations," in *Proceedings of the ACM CoNEXT*, pp. 233–245, New York, NY, USA, December 2019.

[30] I. Cho, K. Jang, and D. Han, "Credit-scheduled delay-bounded congestion control for datacenters," in *Proceedings of the ACM SIGCOMM*, pp. 239–252, New York, NY, USA, August 2017.

[31] K. C. H. W. Li Chen, S. Hu, and D. H. K. Tsang, "Towards minimal-delay deadline-driven data center TCP," in *Proceedings of the ACM HotNets*, pp. 1–7, New York, NY, USA, November 2013.

[32] L. Jose, L. Yan, G. Varghese, and N. McKeown, "Compiling packet programs to reconfigurable switches," in *Proceedings of the USENIX NSDI*, pp. 103–115, Oakland, CA, USA, May 2015.

[33] N. Kr. Sharma, M. Liu, K. Atreya, and A. Krishnamurthy, "Approximating fair queueing on reconfigurable switches," in *Proceedings of the. USENIX NSDI*, pp. 1–16, Renton, WA, USA, April 2018.

[34] Z. Liu, R. B. Basat, G. Einziger et al., "Nitrosketch: robust and general sketch-based monitoring in software switches," in *Proceedings of the. ACM SIGCOMM*, pp. 334–350, New York, NY, USA, August 2019.

[35] S. Hu, W. Bai, G. Zeng et al., "Aeolus: a building block for proactive transport in datacenters," in *Proceedings of the. ACM SIGCOMM*, pp. 1–13, New York, NY, USA, July 2020.

[36] H. Susanto, H. Jin, and K. Chen, "Stream: decentralized opportunistic inter-coflow scheduling for datacenter networks," in *Proceedings of the. IEEE ICNP*, pp. 1–10, Singapore, November 2016.

[37] L. Chen, K. Chen, J. Zhu et al., "Enabling wide-spread communications on optical fabric with MegaSwitch," in *Proceedings of the. USENIX NSDI*, pp. 577–593, Boston, MA, USA, March 2017.

[38] L. Chen, J. Lingys, K. Chen, and F. Liu "AuTO, "Scaling deep reinforcement learning to enable datacenter-scale Automatic traffic optimization," in *Proceedings of the . ACM SIGCOMM*, pp. 191–205, New York, NY, USA, August 2018.

[39] T. Qu, R. Joshi, M. C. Chan, B. Leong, D. Guo, and Z. Liu, "SQR: in-network packet loss recovery from link failures for highly reliable datacenter networks," in *Proceedings of the. IEEE ICNP*, pp. 1–12, Chicago, IL, USA, October 2019.

[40] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," *ACM SIGCOMM - Computer Communication Review*, vol. 42, no. 4, pp. 115–126, 2012.

[41] A. Saeed, V. Gupta, P. Goyal et al., "Annulus: a dual congestion control loop for datacenter and wan traffic aggregates," in *Proceedings of the ACM SIGCOMM*, pp. 735–749, New York, NY, USA, July 2020.

[42] T. Zhang, J. Huang, K. Chen et al., "Rethinking fast and friendly transport in data center networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 5, pp. 2364–2377, 2020.

[43] S. Hu, Y. Zhu, P. Cheng et al., "Tagger: practical PFC deadlock prevention in data center networks," in *Proceedings of the. ACM CoNEXT*, pp. 451–463, New York, NY, USA, November 2017.

[44] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social networks(datacenter) network," *ACM SIGCOMM - Computer Communication Review*, vol. 45, no. 4, pp. 123–137, 2015.

[45] C. Gao, V. C. S. Lee, and K. Li, "D-SRTF: distributed shortest remaining time first scheduling for data center networks," *IEEE Transactions on Cloud Computing*, vol. 9, no. 2, pp. 562–575, 2021.

[46] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, pp. 253–264, New York, NY, USA, December 2012.

[47] A. khurshid, W. Zhou, M. Caesar, P. B. Godfrey, and P. B. Godfrey, "Veri- flow: verifying network-wide invariants in real time," *ACM SIGCOMM - Computer Communication Review*, vol. 42, no. 4, pp. 467–472, 2012.

[48] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: dynamic flow scheduling for data center networks," in *Proceedings of the. USENIX NSDI*, pp. 19–34, San Jose, CA, USA, April 2010.

[49] T. Benson, A. Anand, A. Akella, and M. Zhang, "MicroTE: fine grained traffic engineering for data centers," in *Proceedings of the. ACM CoNEXT*, pp. 1–12, New York, NY, USA, December 2011.

[50] A. Kabbani, B. Vamanan, J. Hasan, and F. Duchene, "Flowbender: flow-level adaptive routing for improved latency and throughput in datacenter networks," in *Proceedings of the. ACM CoNEXT*, pp. 149–160, New York, NY, USA, December 2014.

[51] J. Hu, J. Huang, W. Lv, Y. Zhou, J. Wang, and T. He, "CAPS: codingbased adaptive packet spraying to reduce flow completion time in data center," *IEEE/ACM Transactions on Networking*, vol. 27, no. 6, pp. 2338–2353, 2019.

[52] J. Hu, J. Huang, W. Lyu et al., "Adjusting switching granularity of load balancing for heterogeneous datacenter traffic," *IEEE/ACM Transactions on Networking*, vol. 29, no. 5, pp. 2367–2384, 2021.

[53] J. Liu, J. Huang, W. Li, and J. Wang, "AG: adaptive switching granularity for load balancing with asymmetric topology in data center network," in *Proceedings of the 2019 IEEE 27th International Conference on Network Protocols*, pp. 1–11, Chicago, IL, USA, October 2019.

[54] H. Zhang, J. Zhang, W. Bai, C. Kai, and M. Chowdhury, "Resilient datacenter load balancing in the wild,"pp. 253–266, New York, NY, USA, August 2017.

[55] J. Perry, H. Balakrishnan, and D. Shah, "Flowtune: flowlet control for datacenter networks," in *Proceedings of the USENIX NSDI*, pp. 421–435, Boston, MA, USA, March 2017.

[56] N. Katta, A. Ghag, M. Hira et al., "Clove: congestion-aware load balancing at the virtual edge Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies,"pp. 323–335, New York, NY, USA, November 2017.

[57] P. Wang, H. Xu, Z. Niu, D. Han, and Y. Xiong, "Expeditus: congestion-aware load balancing in Clos data center networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3175–3188, 2017.

[58] N. Katta, M. Hiray, C. Kimz, A. Sivaraman, and J. Rexford, "Hula: scalable load balancing using programmable data planes," in *Proceedings of the Symposium on SDN Research*, p. 10, New York, NY, USA, March 2016.

[59] Z. Guo, Y. Xu, Y. F. Liu et al., "AggreFlow: Achieving Power Efficiency, Load Balancing, and Quality of Service in Data Center Networks," *IEEE/ACM Transactions on Networking*, vol. 29, 2020.

[60] P. Sun, Z. Guo, S. Liu, J. Lan, J. Wang, and Y. Hu, "SmartFCT: Improving Power-Efficiency for Data Center Networks with Deep Reinforcement Learning," *Computer Networks*, vol. 179, 2020.