Adaptive Routing for Datacenter Networks Using Ant Colony Optimization*

Jinbin Hu, Man He, Shuying Rao, Yue Wang, Jing Wang, and Shiming He

School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 410004, China
{jinbinhu,znwj_cs,smhe_cs}@csust.edu.cn
{heman,shuyingrao,wyyy}@stu.csust.edu.cn

Abstract. Modern datacenter networks (DCNs) employ Clos topologies that providing sufficient cross-sectional bandwidth, various load balancing mechanisms are proposed to make full use of multiple parallel paths between end-hosts. Faced with a large number of heterogeneous flows, existing load balancing schemes cannot work well and cause performance degradation, such as latency-sensitive short flows experiencing large tail delay and severe link bandwidth waste due to random rerouting. To solve these issues, we propose an adaptive routing mechanism based on ant colony optimization algorithm (RACO), which adopts different (re)routing strategies for heterogeneous flows. Specifically, RACO uses the improved ant colony optimization algorithm to make optimal rerouting decisions to obtain high throughput and low latency for both elephant flows and mice flows, respectively. The experimental results based on Mininet simulation show that RACO effectively increases the throughput of long flows and reduces the average flow completion time (FCT) of short flows by up to 42% and 61%, respectively, compared with the state-of-the-art load balancing mechanisms.

Keywords: Datacenter networks · Routing · Ant colony optimization · Heterogeneous flows.

1 Inroduction

With the rise of cloud computing and big data, datacenters have become an indispensable part of network computing infrastructure [1–4]. In datacenters, a large number of commercial switches and servers form numerous high-speed clusters through large-scale network interconnection, providing rich computing and storage resources [5, 6]. In order to cope with the increasing bandwidth demand, datacenters typically adopt a tree topology [7] to provide multiple parallel

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

transmission links between nodes, such as Dcell [8], Fat-tree [9], and Bcube [10]. At the same time, various excellent load balancing schemes [11–18] have been proposed to fully utilize multiple parallel paths between terminal hosts.

Load balancing mechanism is one of the important mechanisms to achieve low latency, high bandwidth requirements, and improve network transmission quality in datacenter networks. Through load balancing mechanism, the traffic in datacenter networks can be dispersed across multiple paths, thereby improving the bandwidth utilization and network transmission quality of the network. There are a large number of heterogeneous traffic [19–27] with different transmission requirements[28–30] in datacenter networks, such as latency-sensitive mice flows and throughput-sensitive elephant flows. Therefore, it is crucial for the load balancing mechanism to fully utilize the rich link resources and meet the transmission needs of long and short flows in datacenter networks.

However, existing load balancing mechanisms often encounter problems such as large tail latency and severe link bandwidth waste when facing a large amount of heterogeneous traffic. ECMP [31] is a commonly used standard load balancing mechanism in datacenter networks, which mainly utilizes static hashing mechanism, that is, by utilizing network information to forward traffic to various paths in the network. However, ECMP is vulnerable to hash collisions, in which numerous elephant flows are routed to the same path, causing chain congestion and squandering a significant percentage of the network's available bandwidth. In response to the shortcomings of ECMP, Dixit A et al. proposed the random packet scattering mechanism RPS [32]. It reduces the number of hash conflicts while improving link utilization by randomly sending packets to various available paths in the link. Meanwhile, due to the serious disorder caused by packet granularity transmission, LetFlow [33] based on flowlet transmission has emerged. LetFlow divides the elephant flow by setting time intervals and randomly sends the divided flow to each available path, effectively alleviating the disorder problem and improving link utilization. However, the above methods ignore the transmission characteristics of network traffic and do not adopt reasonable forwarding strategies for heterogeneous flows, resulting in the inability to meet the transmission requirements of throughput-sensitive elephant flows and latency-sensitive mice flows. And due to the randomly selected rerouting method, traffic cannot be accurately routed to the optimal path.

The Ant Colony Optimization (ACO) algorithm, as a heuristic optimization algorithm, has been widely used in network routing [34] and other combinatorial optimization problems [35]. The ant colony optimization algorithm can avoid the congestion problem in datacenter networks through the guidance of pheromone and the path selection strategy, disperse the traffic to different paths, and improve the overall performance and throughput of the network. By using the ant colony optimization algorithm, we can greatly increase the accuracy of traffic routing.

In response to the above analysis, this paper proposes an adaptive routing mechanism called RACO based on ACO algorithm, which selects paths for flows according to their different needs. We improve the heuristic function that guides

ant routing, taking bandwidth utilization and transmission delay as influencing conditions to guide elephant flows in selecting paths with low bandwidth utilization and low latency for mice flows. Through this approach, RACO can effectively improve link utilization and reduce transmission delay, thus meeting the transmission requirements of heterogeneous traffic.

The main contributions of this paper are as follows:

- We conduct in-depth research to analyze the problems caused by ignoring the needs of different types of traffic and randomly selecting rerouted paths: short flows experiencing long tail delays lead to an increase in flow completion time, and random routing of traffic leads to a significant waste of link resources.
- We propose RACO, an adaptive routing mechanism based on ACO algorithm. RACO uses ACO algorithm to select the path rules, redefines the relevant heuristic information, takes the bandwidth utilization and transmission delay as the influencing factors for ants to select the next node, and then selects the optimal transmission path for long and short flows in the network transmission.
- We conduct Mininet simulation under symmetric and asymmetric scenes to test the effectiveness of RACO. The results show that RACO can effectively avoid the long tail congestion and the waste of link resources. Compared with the state-of-the-art schemes, RACO effectively increases the throughput of long flows and reduces the average FCT of short flows by up to 42% and 61%, respectively.

The rest of the paper is organized as following. We describe the background and motivation in Section 2. In Section 3, we describe the overview. In Section 4, we show the Mininet simulation results. In Section 5, we present the related works and then conclude the paper in Section 6.

2 Background and Motivation

2.1 Background

Intelligent optimization algorithms are widely applied in the field of networking[36].The Ant Colony Optimization (ACO) algorithm is an intelligent optimization algorithm that simulates the foraging behavior of ants. At present, it is widely used in path planning, network routing, resource allocation, and other fields. It solves various combinatorial optimization problems by simulating information exchange and cooperation among ants.

The ant colony optimization algorithm is adaptive and robust, and can automatically adapt to the changes and dynamics of the datacenter networks. When the network topology changes, the link fails, or there is a new traffic demand, the ant colony optimization algorithm can adapt to the new situation through the update of pheromones and the adjustment of path selection, without explicit adjustment or re-optimization of the algorithm. The adaptability and robustness of the ant colony optimization algorithm provide significant advantages in dynamic

datacenter network environments. In addition, the ant colony optimization algorithm has relatively low computational and communication overhead. Since the ant colony optimization algorithm only needs each ant to select the path and update the pheromone according to the local information, it does not need the global state information, so the calculation and communication overhead of the algorithm are relatively small. This enables the ant colony optimization algorithm to efficiently perform path selection in datacenter networks with high real-time requirements.

2.2 Motivation

In this section, we first investigate why existing load balancing schemes cannot meet the transmission requirements of heterogeneous traffic in datacenter networks with multiple parallel paths. Then, we conduct extensive simulation experiments to objectively present the performance comparison of existing load balancing schemes when faced with a large amount of heterogeneous traffic transmission.

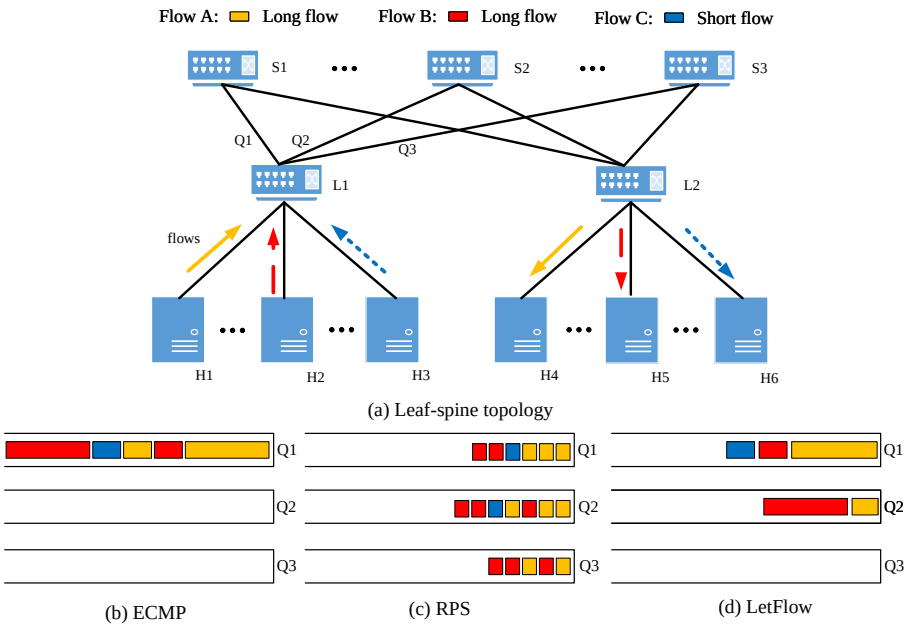


Fig. 1. Queueing under different load balancing mechanisms

Problem Description. In datacenter networks, blindly selecting rerouting paths without considering the needs of different traffic may cause damage to net-

work transmission performance. We choose three typical load balancing schemes, ECMP, RPS, and LetFlow, to illustrate this issue and demonstrate its impact.

We show how ECMP, RPS, and LetFlow work in Figure 1. In Figure 1(a), we have sent three flows from the senders (H1, H2, H3) to the receivers (H4, H5, H6), consisting of two long flows (A, B) and one short flow C. These flows from switch L1 arrive at switch L2 using three parallel paths, each aligned with the three output port queues (Q1, Q2, Q3) of switch L1.

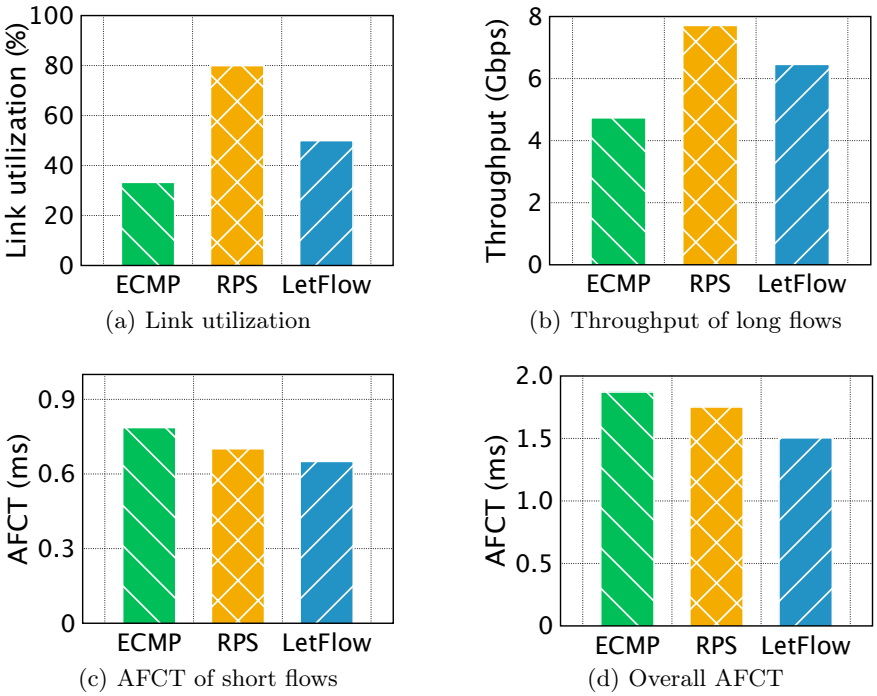
ECMP selects a fixed forwarding path for the flow based on the five tuples, so it is inevitable that different flows will be clustered on the same path, resulting in long tail latency and hash conflicts. As illustrated in Figure 1(b), it can be seen that short flow C ranks behind long flows (A, B), while long flows (A, B) are all routed to a single path, and the paths corresponding to the remaining queues (Q2, Q3) are not fully utilized. The RPS scheme effectively improves link utilization by randomly forwarding packets to various available paths. However, due to unknown path conditions, it may lead to issues such as long tail delay and disorder. As shown in Figure 1(c), the short flow C is still randomly placed after the long flows (B, C), and there are out-of-order packets. LetFlow distinguishes grouping clusters based on time interval thresholds and randomly sends them to other paths. Due to the randomness of rerouting, it may lead to conflicts between short and long flows. As shown in Figure 1(d), short flow C was randomly rerouted to the end of the long flow, experiencing a long tail delay, while the path corresponding to queue Q3 was not fully utilized.

Performance Impact. To more intuitively demonstrate the network performance of existing load balancing schemes in the face of large amounts of heterogeneous traffic transmission, we conducted a series of NS-3 simulation tests. We use a 2×10 leaf-spine topology providing 10 equivalent paths between hosts. The bandwidth of each path is 10Gbps, and the round-trip propagation delay is 100 μs. The size of the switch buffer is set to 200 packets, and the flow timeout is set to 150 μs. In the experimental test, we randomly generate DCTCP flows according to the heavy-tailed distribution. Through experimental simulations, we systematically compared the link utilization, throughput of long flows, average completion time of short flows, and average completion time of all flows for ECMP, RPS, and LetFlow.

Link resources are very expensive in datacenter networks, and whether to fully utilize link resources is one of the important criteria for evaluating the quality of a load balancing mechanism. As shown in Figure 2(a), the link utilization under various load balancing mechanisms varies. Among them, ECMP based on flow granularity transmission has the lowest link utilization rate (only one-third), which is due to the adverse consequences of a large number of hash collisions leading to multiple flows be sent to the same path for transmission. Although LetFlow has improved performance compared to ECMP, the use of flowlet granularity to randomly select rerouted paths still results in low link utilization. In addition, although the link utilization of RPS transmitted in packet

granularity is significantly higher than the other two schemes, it only reaches 80%, and there is still a utilization inefficiency in link resources.

As shown in Figure 2(b), consistent with the comparison of link utilization, the throughput of ECMP's long flows is still at its lowest level. ECMP is prone to forwarding different long flows to the same path, leading to network congestion and reducing the throughput of long flows. We can also observe that the throughput of long flows with high link utilization RPS is higher than that of ECMP and LetFlow. This is because RPS can divide long flows into multiple data packets for transmission on different paths, improving the throughput of long flows. But it is precisely because of the random transmission of multiple packets that the problem of out-of-order retransmission caused by RPS is very serious and has a negative impact.



(a) Link utilization (b) Throughput of long flows

(c) AFCT of short flows (d) Overall AFCT

Fig. 2. Performance under different load balancing mechanisms

As shown in Figure 2(c) and Figure 2(d), LetFlow has significantly lower average completion time of short flows and average FCT for overall flows compared to ECMP and RPS. This is because ECMP may forward both long and short flows to the same path, causing short flows to experience significant long tail delays, thereby increasing the average flow completion time of short flows. The waste of link resources and the low throughput of long flows fully validate

the fact that the average FCT of overall flows of ECMP is high. At the same time, RPS randomly sends packets to various paths through the switch, which can lead to adverse effects such as disorderly retransmission or even packet loss retransmission of each data packet, resulting in an increase in FCT and affecting network transmission performance.

2.3 Summary

Therefore, based on the observations of the above experiments, we conclude that (1) random rerouting leads to the failure to fully utilize the resources of each link, resulting in serious bandwidth waste and low link utilization, thereby affecting the throughput of long flows. (2) Random rerouting results in long and short flows sharing the same path, and short flows experience large tail delays, increasing the completion time of short flows. We propose an adaptive routing mechanism based on ACO algorithm, which selects different forwarding paths for long and short flows through the routing rules of ACO algorithm to meet the different transmission requirements of long and short flows. We introduce our design details in the rest of this paper.

3 RACO Design

3.1 RACO Overview

The RACO mechanism proposed in this paper uses the rules of the ant colony optimization algorithm to select the path to balance the load in datacenter networks. The basic idea is to abstract the traffic scheduling problem in datacenter networks into a graph theory problem and regard the datacenter networks as the weighted graph, the switch or host is regarded as the node of the weighted graph, and the transmission link between the switch and the host is represented as the corresponding edge in the weighted graph. At the same time, the transfer of traffic is regarded as the path from the source node to the destination node. Specifically, the path taken by the ant colony can be likened to the viable path for traffic scheduling in the datacenter network, and the search space of the ant colony optimization algorithm is the entire network link.The information that guides the ants in their path selection can be likened to the network status information, which is continuously updated throughout the iterative process. As the number of iterations increases, the optimal path in the network will be covered by more pheromones, which greatly increases the possibility of the ant colony choosing the optimal path. Promoted by positive feedback, the ant colony optimization algorithm can effectively point to the best path in the network, so as to realize the effective scheduling of convection. RACO consists of two modules, as shown in Figure 3.

(1) **Statistics Module.** In RACO, we first distinguish between traffic types, dividing the flow into elephant and mice flows based on the number of bytes sent by the flow. Then according to the different requirements of two kinds of traffic, the relevant network state information is collected.

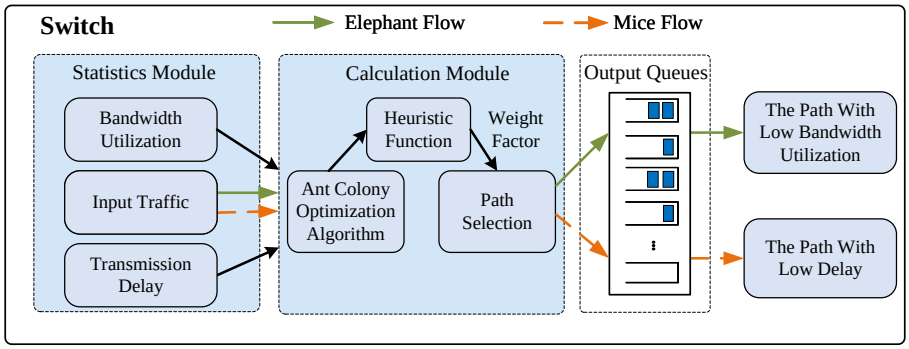


Fig. 3. RACO architecture

(2) **Calculation Module.** The calculation module is mainly responsible for deciding different forwarding strategies based on the flow size. Because of the need to ensure as fast as possible, the mice flow chooses the path with lower delay to avoid long tail delay. The long flow selects the path with low bandwidth utilization to make full use of the abundant link resources.

3.2 Statistics Module

The adaptive routing mechanism based on ACO algorithm proposed in this paper requires distinguishing between elephant flows and mice flows, and implementing different path selection strategies to meet the requirements of high bandwidth for elephant flows and low latency for mice flows. So this paper distinguishes traffic based on the number of bytes sent by the flow. When the number of bytes received exceeds the set threshold (set to 100KB in this paper), it is determined that the sent flow is an elephant flow; Otherwise, it's a mice flow. In addition, we use graph to represent the topology of datacenter network. A set S= $\{s1, s2, ..., s_n\}$ is used to represent a switch node, and the set L= $\{l1, l2, ..., l_n\}$ represents a link in the network, where l_{ij} is represented as a link between l_i node and l_j node.

In order to meet the needs of high throughput of elephant flows and low latency of mice flows in datacenter networks, we need to divide the statistical network state information into transmission delay and bandwidth utilization, because it has a great impact on the performance of long and short flows. We choose $band(l_{ij})$ to represent inter-link bandwidth utilization and $delay(l_{ij})$ represent inter-link transmission delay. in datacenter networks, we take ants as packets of data. We record the network state information through the switch, which is used as the influence factor for the ants to choose the available path for different types of flows. We maintain a link state estimation table on the switch port that records the addresses, bandwidth utilization, and transmission delays of all the next switch nodes. We use the method used in hula to estimate

the bandwidth utilization of the link. In addition, we record the transmission delay by sending the probe packet. When the probe packet passes through each switch, each switch records its sending time. When the probe packet reaches the receiver and generates an ACK, time the ACK was generated. By subtracting the time it takes the switch to record the probe packet's send time from the time it produces the ACK, we get the delay for each path. Of course, the transmission path of the probe packet and ACK may be different, and we can send multiple probe packets simultaneously to ensure that the transmission delay is measured. In order to ensure the validity of the network state, we set a fixed time interval (the default setting is 500 μs) to update the state information.

In addition, we use the method used in [37] to estimate the bandwidth utilization of the link:

$$U = D + U \times (1 - \frac{\Delta t}{\tau}) \tag{1}$$

where U is the estimated bandwidth utilization of the link, D represents the size of the data packet, Δt represents the time elapsed since the last update of bandwidth utilization, τ is a time constant and should be set to at least twice the sending frequency of the probe packet.

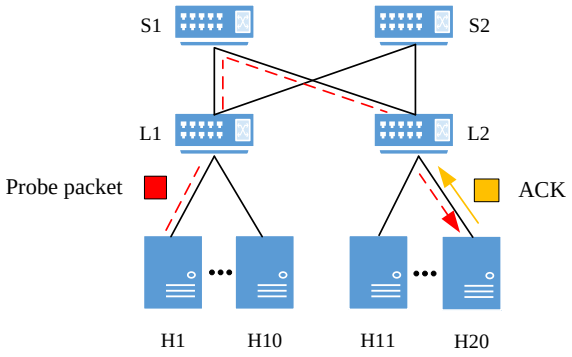


Fig. 4. Calculation of transmission delay

As shown in the Figure 4, we use a 2×2 leaf-spine topology to illustrate how to measure the transmission delay. We send a probe packet from H1 to H2. When passing through switches L_1, S_1, L_2, we record the sending time of this probe packet on each switch. When the probe packet arrives at H2 to generate ACK, we record the time of generating ack. In this way, when the switch receives the ACK, we can subtract the sending time of each switch's recorded probe packet from the time of generating the ack to obtain the path delay from each switch to the destination node. Then we can subtract the path delay from

two adjacent switches to the destination node to obtain the path delay of the adjacent switch.For example, $delay(S_1-L_2) = delay(S1-H2)-delay(L_2-H_2)$.

3.3 Calculation Module

The calculation module mainly makes different path calculation decisions for elephant flows and mice flows to meet their different transmission needs. The rule for calculating the path selection is based on the transition probability P_{ij} to select the next node. The calculation formula is shown in equation (2):

$$P_{ij} = \begin{cases} \frac{(\tau_{ij})^{\alpha} \times (\eta_{ij})^{\beta}}{\sum_{u \epsilon ad_k} (\tau_{iu})^{\alpha} \times (\eta_{iu})^{\beta}}, & u \in ad_k \\ 0, & others \end{cases} \tag{2}$$

Among them, P_{ij} represents the probability of ants moving from node i to node j, ad_k represents k paths that ants can move, and α represents the importance of pheromone. β represents the importance of the heuristic factor, τ_{ij} represents the concentration of pheromone on path l_{ij}, η_{ij} refers to heuristic function on path l_{ij}.

We can see from the above formula that the transition probability P_{ij} that affects the ant to select the next node is mainly affected by pheromone concentration and heuristic function. In this paper, we mainly improve the heuristic function to meet the needs of long and short flows. Specifically, considering the impact of transmission delay and bandwidth utilization on traffic transmission, we propose a corresponding improved heuristic function to guide ants to select the next node. Path l_{ij} of the heuristic function on is shown in equation (3):

$$\eta = \frac{\varepsilon}{band(l_{ij})} + \frac{\theta}{delay(l_{ij})} \tag{3}$$

Among them, ε and θ are weight factors, $0 < \varepsilon < 1$, $0 < \theta < 1$, and the values of varepsilon and theta are determined based on the current type of traffic. When the forwarded traffic in the network is elephant flow, it is more sensitive to link bandwidth utilization, and the value of ε should be greater than θ. On the contrary, when the traffic forwarded in the network is mice flow, it requires a higher transmission delay of the link, and the value of θ is greater than ε. The larger the $band(l_{ij})$ and $delay(l_{ij})$ values, the higher the load and delay of this link. Therefore, the shorter the value of the heuristic function, the less likely the ant is to choose the path, in order to guide the ant in heuristic routing.

In addition, in order to ensure the accuracy of the algorithm, we will adjust the pheromone of the path the ants go through to obtain the optimal result after the ants complete a routing. The update of pheromone can be realized by the following formula:

$$\tau_{ij}(t+1) = (1-P) \times \tau_{ij}(t) + \nabla \tau_{ij}(t) \tag{4}$$

Among them, $\tau_{ij}(t+1)$ is the pheromone concentration on the path l_{ij}, $\nabla\tau_{ij}(t)$ is the increment of pheromone on the path l_{ij}, and P is the volatilization factor of pheromone.

When the ant reaches its destination, we record its path in the Path Table PathList. The PathList= $[P1, P2, ..., P_n]$. We use the SelectBestPath function to select the most suitable path for mice and elephant flows. Specifically, when the next transmission of the data flow for the mice flow, we can choose the optimal path Ph in the path of the least delay. When the next data flow is an elephant flow, the path with the lowest bandwidth utilization in the optimal path Ph is chosen. In addition, RACO updates the optimal path at regular intervals (the default is 500 μs) to ensure the efficiency of the flow.

3.4 Algorithm Pseudocode

This paper proposes an adaptive routing mechanism based on ACO algorithm, which improves the heuristic function according to the different needs of long and short flows. By allocating size and network state information through weight factors, the number of link congestion is reduced, throughput and link utilization are improved. The RACO algorithm processing pseudocode is as follows:

Algorithm 1: RACO Algorithm

Input: The first node on the path *src*, the last node on the path *dst*,$G(S, L)$
Output: The optimal path Ph

```
while iter < M do
    for i = 1 to k do
        Set curNode=src;
        while curNode! = src do
            Choose next node with P_ij;
            path.append(curNode);
        end
        PathList(path) update τ_ij
    end
end
Ph=SelectBestPath(PathList);
return Ph
```

4 Performance Evaluation

4.1 Experimental Setup

Choices of Basedline. We choose four load balancing mechanisms, ECMP, RPS, DRILL, and LetFlow, to compare with RACO. ECMP is a common load balancing mechanism in datacenter networks, which distributes flows to each

path by hashing. RPS is based on packets, which are sent randomly to each available path. Drill selects the port with the smallest queue length as the packet forwarding port by comparing the two currently randomly selected ports with the one with the smallest load in the previous round. LetFlow divides the flow according to a set time interval and randomly selects a forwarding port for the divided flow.

Simulation Setup. We build a K=4 Fat-tree topology using Mininet network simulation platform. The topology consists of 20 switches, 16 hosts and 48 links. In all experiments, we set the link bandwidth to 1 Gbps, the link propagation delay to 100μs, and the buffer size of the switch to 256 packets. All traffic generation follows a Poisson distribution. In addition, we adjusted the workload from 0.4 to 0.7 to evaluate RACO performance.In order to reduce the adverse effect of random error and guarantee the fairness of the experiment, We test each load balancing mechanism twenty times, and take the average of twenty times as the final result.

At the same time, the parameter settings of the RACO algorithm also affect its performance. Based on multiple simulation tests, this paper sets the following parameters, as shown in Table 1.

Table 1. RACO algorithm parameters.

Parameter	Value
Iterations M	10
Pheromone volatile factor p	0.2
Pheromone weight α	2
Heuristic factor β	3
weight factors ε	mice flows: 0.3, elephant flows: 0.7
weight factors θ	mice flows: 0.7, elephant flows: 0.3

4.2 Performance Under Symmetrical Topology

In order to evaluate the effectiveness of the adaptive routing mechanism RACO based on ACO algorithm, link utilization, throughput, round-trip delay and average FCT are tested and analyzed under different network loads in a symmetric network topology, in conjunction with ECMP, RPS, DRILL, and LetFlow. The experimental results are as follows:

As shown in Figure 5, RACO consistently maintains the highest link utilization compared to ECMP, RPS, DRILL, and LetFlow as network load increases. As a whole, with the increasing network load, the link utilization of all load balancing mechanisms is increasing, but the growth rate is different. For example, with a network load of 0.6, RACO has increased link utilization by about 12%, 14%, 26%, compared to DRILL, LetFlow, and ECMP, respectively. This

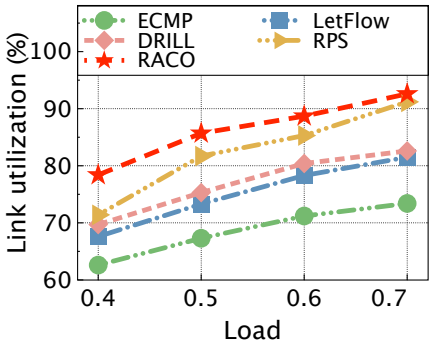


Fig. 5. Link utilization

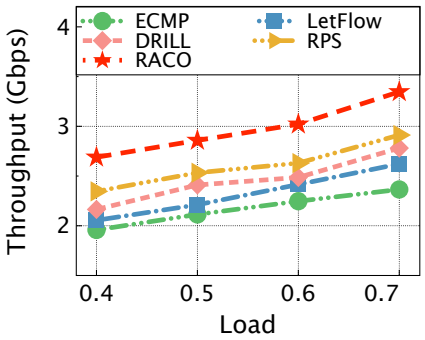


Fig. 6. Throughput of long flows

is because RACO can choose the path with low bandwidth utilization for elephant flows, avoid the waste of link resources, and improve link utilization. RPS and DRILL based on packet granularity also have relatively high link utilization, while ECMP has the lowest link utilization under different loads. This is because ECMP uses hash mechanism to schedule flows, different flows may choose the same fixed path forwarding, easy to cause a link in idle state for a long time.

Figure 6 shows that RACO consistently maintains the highest throughput compared to other advanced load balancing mechanisms as the network load increases. Specifically, RACO increases throughput by about 34%, 15%, 22%, and 33%, compared to ECMP, RPS, DRILL, and LetFlow at network load of 0.6. This is because RACO can route elephant flows to low bandwidth utilization links based on heuristic information, which greatly improves link utilization and overall throughput. The flow-based load balancing routing strategy, ECMP, will lead to the waste of link resources. Due to the inevitable hash collision and the design flaw of not being able to sense the congestion, which will also aggravate the congestion of the link, this will reduce network throughput.

As shown in Figure 7, the round trip delay increases as the load increases, but the round trip delay of RACO is always at the lowest level. Specifically, at a network load of 0.6, RACO reduces round-trip latency by 35%, 21%, 16%, and 8% compared to ECMP, RPS, DRILL, and LetFlow, respectively. This is because RACO distinguishes heterogeneous traffic and routes elephant traffic to transport paths with low link utilization. This effectively equalizes the transmission pressure of each path in the network, avoids the link congestion, and completes the transmission faster. In addition, RACO makes the mice flow always route to the switch port with the lowest delay, effectively reducing the probability of long and short flows routing together, avoiding the long tail delay of short flows, and effectively reducing the transmission delay.

Figure 8 shows that RACO significantly reduces the average FCT as network load increases compared to ECMP, RPS, DRILL, and LetFlow. The performance of fine-grained solutions such as RPS and DRILL decreases as the network load

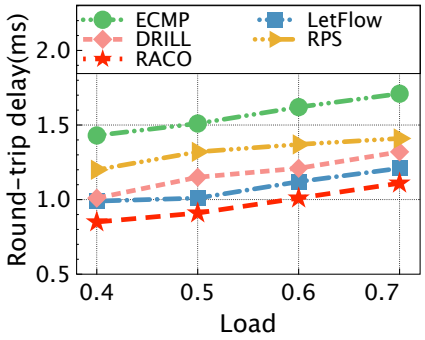


Fig. 7. Round-trip delay

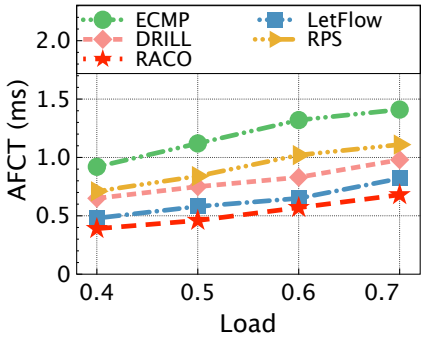


Fig. 8. AFCT of all flows

increases. This is because the congestion path increases and the packet disorder becomes more serious, causes more packet retransmissions to occur. However, ECMP and Letflow, as coarse-grained schemes, do not fully utilize the link resources in the network and do not take into account the transmission demand of heterogeneous traffic. So the average FCT is higher. RACO takes into account the needs of different traffic, and combined with the ACO algorithm iterative fast characteristics, can be more accurate and faster for the flow to find the best link path condition. More specifically, compared to ECMP, RPS, DRILL, and LetFlow at a load of 0.6, RACO can reduce the average FCT by 56%, 45%, 31%, and 12%, respectively.

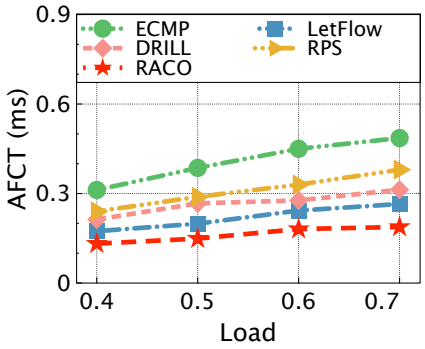


Fig. 9. AFCT of short flows

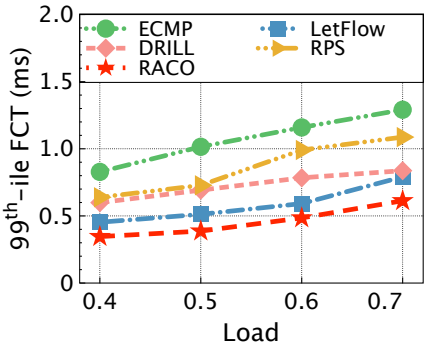


Fig. 10. 99^{th}-ile FCT of short flows

Figure 9 and Figure 10 show the mean FCT and 99^{th} percentile FCT of the short flow from load 0.4 to 0.7, respectively. We notice that RACO is able to effectively reduce AFCT and tail FCT compared with the remaining four schemes. Specifically, at network load of 0.7, RACO reduces ACT and 99^{th} percentile FCT

by 61%, 50%, 40%, 29% and 53%, 44%, 27%, 23% compared to ECMP, RPS, DRILL, and LetFlow, respectively. This is because when the workload becomes larger, more short flows will experience queuing delay and out-of-order retransmission. RACO can select the path with low transmission delay for mice flows, effectively improving the transmission performance of short flows and reducing queuing delays and the impact of retransmissions.

4.3 Performance Under Asymmetric Topology

In order to evaluate the effectiveness of the adaptive routing mechanism RACO based on ACO algorithm, link utilization, throughput, round-trip delay and average FCT are tested and analyzed under different network loads in an asymmetric network topology, in conjunction with ECMP, RPS, DRILL, and LetFlow.

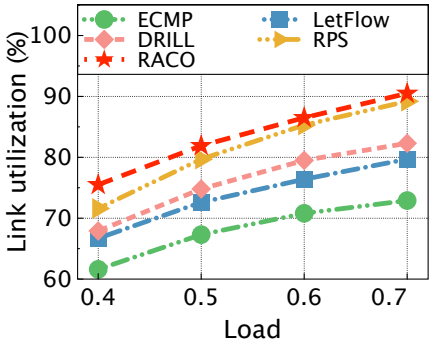


Fig. 11. Link utilization

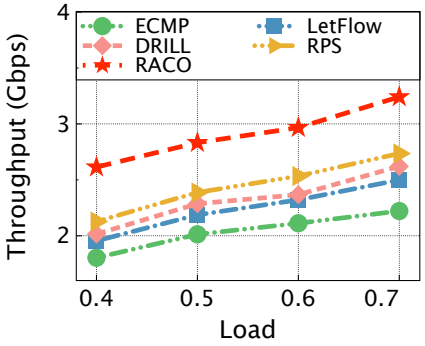


Fig. 12. Throughput of long flows

As shown in Figure 11, RACO consistently maintains the highest link utilization compared to ECMP, RPS, DRILL, and LetFlow as network load increases. Specifically, RACO increases link utilization by 34%, 15%, 22%, and 33%, compared to ECMP, RPS, DRILL, and LetFlow at a load of 0.6. This is because RACO makes good use of the idle links, and disperses the transmission pressure of each link in the network, so as to avoid the extreme situation that one link is always hungry while causing congestion. In this way, RACO can route the elephant flows to the idle link according to the heuristic information even in the asymmetric network transmission scenario, which makes full use of the link resources and improves the link utilization.

Figure 12 shows that RACO consistently maintains the highest throughput compared to other advanced load balancing mechanisms as the network load increases. Specifically, with a network load of 0.7, RACO increases throughput by approximately 46%, 19%, 24%, and 30%, compared to ECMP, RPS, DRILL, and LetFlow, respectively. RACO searches and optimizes continuously through heuristic information, and makes rational use of link resources to a large extent.

Even under the fast changing network load, elephant flows can still be sent quickly to the optimal forwarding path. Therefore, in an asymmetric topology, RACO still performs well with increasing load. Although RPS can improve link utilization and throughput by randomly scattering packets. The serious out-of-order problem can not be ignored.

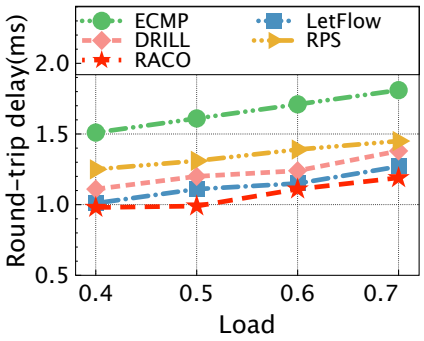


Fig. 13. Round-trip delay

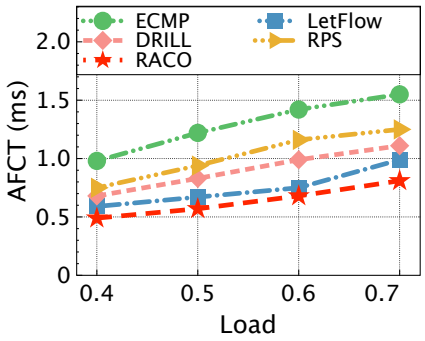


Fig. 14. AFCT of all flows

As shown in Figure 13, RACO always maintains the lowest round trip latency compared to other advanced load balancing mechanisms under different network loads. This is because RACO fully considers the transport characteristics of heterogeneous traffic, routes the throughput-sensitive elephant flows to the port with low link utilization while routing the delay-sensitive mice flows to the port with low delay. This design can improve the link utilization and reduce the probability of long tail delay, thus reducing the round trip delay. In the asymmetric network scenario, the round trip delay of ECMP and RPS, which are prone to long tail delay and serious packet reorder. Specifically, compared with ECMP, RPS, DRILL, and LetFlow, the round trip delay is decreased by 35%, 20%, 10%, and 9%, respectively, at the load of 0.6.

Figure 14 shows that RACO significantly reduces the AFCT compared to ECMP, RPS, DRILL, and LetFlow as the network load increases. Specifically, at the load of 0.6, RACO reduced the average FCT by 52%, 41%, 31%, and 9% compared with ECMP, RPS, DRILL, and LetFlow, respectively. The results show that RACO still performs well in asymmetric topology. This is because RACO can continuously search and optimize by heuristic information, and can quickly adapt to network load changes, and choose the appropriate forwarding path for the flows in time. Through reasonable path forwarding, RACO can make full use of link resources to avoid long tail delay, thus reducing the average FCT.

As shown in Figure 15 and Figure 16, under different network loads, compared with other advanced load balancing mechanisms, the average completion

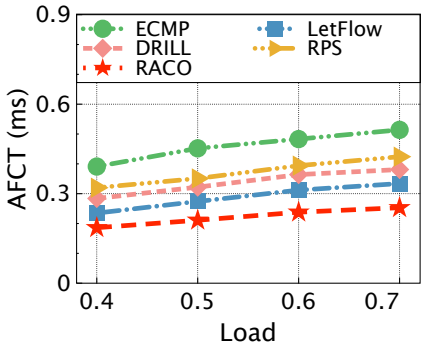


Fig. 15. AFCT of short flows

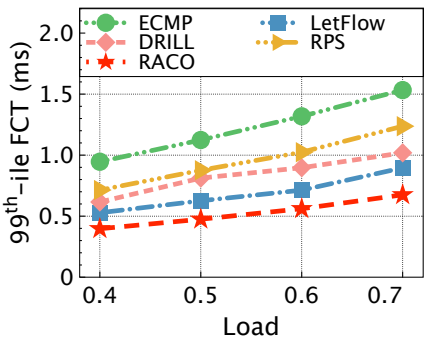


Fig. 16. 99^{th}-ile FCT of short flows

time and 99^{th} percentile FCT of short flows in RACO always maintain a lower level. Specifically, when the network load is 0.7, RACO reduces the average flow completion time of short flows by about 51%, 40%, 33% and 32% compared with ECMP, RPS, DRILL and LetFlow, respectively. The 99^{th} percentile FCT dropped by about 56%, 45%, 34%, and 25%. These test results prove that RACO still has advantages in asymmetric topologies. This is because RACO can dynamically select a more appropriate forwarding path for short flows based on the heuristic information, thereby avoiding long-tail delays for short flows and improving the transmission quality of short flows.

5 Related Work

Load Balancing Schemes Based on Flow. ECMP is a load balancing strategy based on flow granularity. In order to increase network capacity usage, a router will distribute various types of traffic to various links when it finds many equivalent paths that lead to the same destination address. ECMP, on the other hand, is unable to detect congestion, and for lines that are already congested, it is likely to make the situation worse. Hedera [38] achieves effective transmission of elephant flows by navigating all open channels on the network to identify the first one that can match user bandwidth requirements. Although this approach reduces network congestion, the chosen path might not be the best one.Mahout [39] monitors host side flows for elephant fluxes, which are then identified and tagged. The controller assigns the elephant flows to lighter loaded channels once the switch node gets the identified elephant flows. Although it requires changes to the host, this technique can significantly lower the switch's overhead.

Load Balancing Schemes Based on Flowlet. DRE technology is primarily used by CONGA [40] to gauge and assess the level of obstruction in the path. The switch keeps a brief flow information table and a congestion information table, and the switch chooses the flow's forwarding path based on the flow and

congestion information. Adopting global congestion aware technology can reduce link latency and increase resource utilization, but it still has difficult deployment problems because it needs to store a lot of path information on switches. LetFlow makes use of the inherent properties shared by packets to detect path congestion automatically. Using time interval criteria to detect packet clusters, it sends them to different pathways at random. LetFlow can prevent chaos and successfully handle asymmetric issues. However, due to the randomness of LetFlow scheduling, optimal load balancing performance cannot be achieved.

Load Balancing Schemes Based on Packet. An approach for load balancing at the packet level is RPS. In this approach, multiple equivalent pathways to the same destination address are used for each connection on a per-packet basis, as determined by the router.Although it is straightforward, simple to deploy, and fully utilizes network links, it may cause major disorder issues. Between two random ports and the port with the least queue length from the previous round, DRILL [41] chooses a forwarding path. The basic idea behind path selection is to choose the port closest to them with the shortest wait length, then use that way to transmit the packet. However, because it may assess the status of the path based on local information, it is unable to completely eliminate the disorder issue.

6 Conclusion

In this paper, we propose RACO, an adaptive routing mechanism based on ACO algorithm in datacenter networks. RACO uses ACO algorithm to find the optimal path, and uses bandwidth utilization and transmission delay as heuristic information for ant to select the path. Then, the long and short flows are rerouted to the transmission path with low link utilization and low delay respectively, so as to avoid encountering the path with congestion and improve the network transmission performance. According to the Mininet simulation results, RACO can effectively avoid the long tail congestion and the waste of link resources. Compared with the most advanced solutions, RACO effectively increases the throughput of long flows and reduces the average FCT of short flows by up to 42% and 61%, respectively. In the future work, we plan to test the performance of RACO in a variety of different real scenarios and further improve the mechanism.

References

1. Wei, W., Gu, H., Wang, K., et al.: Multi-dimensional resource allocation in distributed data centers using deep reinforcement learning. IEEE Transactions on Network and Service Management. **20**(2), 1817-1829 (2022)
2. Li, H., Zhang, Y., Li, D., et al.: Ursa: Hybrid block storage for cloud-scale virtual disks. In: Proceedings of the Fourteenth EuroSys Conference, pp. 1-17 (2019)
3. Zhao, Y., Huang, Y., Chen, K., Yu, M., et al.: Joint VM placement and topology optimization for traffic scalability in dynamic datacenter networks. Computer Networks. **80**, 109-123 (2015)

4. Wang, J., Yuan, D., Luo, W., et al.: Congestion control using in-network telemetry for lossless datacenters. Computers, Materials & Continua. **75**(1), 1195-1212 (2023)
5. Wang, Y., Wang, W., Liu, D., et al.: Enabling edge-cloud video analytics for robotics applications. IEEE Transactions on Cloud Computing. **11**(2), 1500-1513 (2023)
6. Hu, J., Huang, J., Li, Z., Wang, J., He, T.: A Receiver-Driven Transport Protocol with High Link Utilization Using Anti-ECN Marking in Data Center Networks. IEEE Transactions on Network and Service Management. **20**(2), 1898-1912 (2023)
7. Zheng, J., Du, Z., Zha, Z., et al.: Learning to Configure Converters in Hybrid Switching Data Center Networks. IEEE/ACM Transactions on Networking. 1-15 (2023)
8. Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., Lu, S.: Dcell: a scalable and fault-tolerant network structure for data centers. In: Proceedings of ACM SIGCOMM, pp. 75-86 (2008)
9. Al-Fares, M., Loukissas, A., Vahdat, A.: A scalable, commodity data center network architecture. ACM SIGCOMM computer communication review. **38**(4), 63-74 (2008)
10. Guo, C., Lu, G., Li, D., et al.: BCube: a high performance, server-centric network architecture for modular data centers. In: Proceedings of ACM SIGCOMM, pp. 63-74 (2009)
11. Hu, J., Huang, J., Lv, W., Zhou, Y., Wang, J., He, T.: CAPS: Coding-Based Adaptive Packet Spraying to Reduce Flow Completion Time in Data Center. In: Proceedings of IEEE INFOCOM, pp. 2294-2302 (2018)
12. Hu, J., Huang, J., Lv, W., Li, W., Wang J., He, T.: TLB: Trafficaware Load Balancing with Adaptive Granularity in Data Center Networks. In: Proceedings of ACM ICPP, pp. 1-10 (2019)
13. Hu, J., He, Y., Wang, J., et al.: RLB: Reordering-Robust Load Balancing in Lossless Datacenter Network. In: Proceedings of ACM ICPP (2023)
14. Hu, J., Zeng, C., Wang, Z., et al.: Enabling Load Balancing for Lossless Datacenters. In: Proceedings of IEEE ICNP (2023)
15. Zhang, H., Zhang, J., Bai, W., Chen, K., Chowdhury, M.: Resilient datacenter load balancing in the wild. In: Proceedings of ACM SIGCOMM, pp. 253–266 (2017)
16. Liu, Y., Li, W., Qu, W., Qi, H.: BULB: Lightweight and Automated Load Balancing for Fast Datacenter Networks. In: Proceedings of ACM ICPP, pp. 1-11 (2022)
17. Hu, J., Zeng, C., Wang, Z., Xu, H., Huang, J., Chen, K.: Load Balancing in PFC-Enabled Datacenter Networks. In: Proceedings of ACM APNet (2022)
18. Xu, R., Li, W., Li, K., Zhou, X., Qi, H.: DarkTE: Towards Dark Traffic Engineering in Data Center Networks with Ensemble Learning. In: Proceedings of IEEE/ACM IWQOS, pp. 1-10 (2021)
19. Li, W., Chen, S., Li, K., Qi, H., Xu, R., Zhang, S.: Efficient online scheduling for coflow-aware machine learning clusters. IEEE Transactions on Cloud Computing. **10**(4), 2564-2579 (2020)
20. Wang, J., Rao, S., Liu, Y., et al.: Load balancing for heterogeneous traffic in datacenter networks. Journal of Network and Computer Applications. 217 (2023)
21. Wei, W., Gu, H., Deng, W., et al.: ABL-TC: A lightweight design for network traffic classification empowered by deep learning. Neurocomputing. **489**, 333-344 (2022)
22. He, X., Li, W., Zhang, S., Li, K.: Efficient Control of Unscheduled Packets for Credit-based Proactive Transport. In: Proceedings of ICPADS, pp. 593-600 (2023)

23. Li, W., Yuan, X., Li, K., Qi, H., Zhou, X.: Leveraging endpoint flexibility when scheduling coflows across geo-distributed datacenters. In: Proceedings of IEEE INFOCOM, pp. 873-881 (2018)
24. Hu, C., Liu, B., Zhao, H., Chen, K., et al.: Disco: Memory efficient and accurate flow statistics for network measurement. In: Proceedings of IEEE ICDCS, pp. 665-674 (2010)
25. Bai, W., Chen, K., Hu, S., Tan, K., Xiong, Y.: Congestion control for high-speed extremely shallow-buffered datacenter networks. In: Proceedings of ACM APNet, pp. 29-35 (2017)
26. Cho, I., Jang, K., Han, D.: Credit-scheduled delay-bounded congestion control for datacenters. In: Proceedings of ACM SIGCOMM, pp. 239-252 (2017)
27. Hu, C., Liu, B., Zhao, H., et al.: Discount counting for fast flow statistics on flow size and flow volume. IEEE/ACM Transactions on Networking. **22**(3), 970-981 (2013)
28. Li, Z., Bai, W., Chen, K., et al.: Rate-aware flow scheduling for commodity data center networks. In: Proceedings of IEEE INFOCOM, pp. 1-9 (2017)
29. Zhang, J., Bai, W., Chen, K.: Enabling ECN for datacenter networks with RTT variations. In Proceedings of ACM the 15th International Conference on Emerging Networking Experiments And Technologies, pp. 233-245 (2020)
30. Wang, J., Liu, Y., Rao, S., et al.: Enhancing security by using GIFT and ECC encryption method in multi-tenant datacenters. Computers, Materials & Continua. **75**(2), 3849-3865 (2023)
31. Hopps, C.E.: Analysis of an equal-cost multi-path algorithm (2000)
32. Dixit, A., Prakash, P., Hu Y. C., Kompella R. R.: On the impact of packet spraying in data center networks. In: Proceedings of IEEE INFOCOM, pp. 2130–2138 (2013)
33. Vanini, E., Pan, R., Alizadeh, M., Taheri, P., Edsall, T.: Let it flow: Resilient asymmetric load balancing with flowlet switching. In: Proceedings of NSDI, pp. 407–420 (2017)
34. Lv, J., Wang, X., Ren, K., Huang, M., Li, K.: ACO-inspired information-centric networking routing mechanism. Computer Networks. 126, 200-217 (2017)
35. Gupta, A., Garg, R.: Load balancing based task scheduling with ACO in cloud computing. In: Proceedings of IEEE ICCA, pp. 174-179 (2017)
36. Wang, J., Liu, Y., Rao, S., et al.: A novel self-adaptive multi-strategy artificial bee colony algorithm for coverage optimization in wireless sensor networks. Ad Hoc Networks. **150**,(2023)
37. Katta, N., Hira, M., Kim, C., Sivaraman, A., Rexford, J.: Hula: Scalable load balancing using programmable data planes. In: Proceedings of ACM SOSR, pp. 1-12 (2016)
38. Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., Vahdat, A.: Hedera: dynamic flow scheduling for data center networks. In: Proceedings of NSDI, pp. 89-92 (2010)
39. Curtis, A. R., Kim, W., Yalagandula, P.: Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection. In: Proceedings of IEEE INFOCOM, pp. 1629-1637 (2011)
40. Alizadeh, M., Edsall, T., Dharmapurikar, S., et al.: CONGA: Distributed congestion-aware load balancing for datacenters. In: Proceedings of ACM SIGCOMM, pp. 503–514 (2014)
41. Ghorbani, S., Yang, Z., Godfrey, P. B., et al.: DRILL: Micro load balancing for low-latency data center networks. In: Proceedings of ACM SIGCOMM, pp. 225–238 (2017)