Elephant Flow Scheduling in SDN Data Center Networks Based on Ant Colony Algorithm: A Postprint

Authors: Li Honghui, Yang Guang, Lu Hailiang, Fu Xueliang, Shen Zhijun

Date: 2018-10-11T00:00:00+00:00

Abstract

To address the problems of data center network congestion and load imbalance caused by traditional methods when scheduling elephant flows, we propose ACO-SDN, an SDN (Software-Defined Networking) data center network traffic scheduling algorithm based on ant colony optimization. An Integer Linear Programming (ILP) model is established for the elephant flow scheduling problem, with the optimization objective of minimizing the maximum link utilization. The ILP model is solved by redefining the parameters and operations of the ant colony algorithm to obtain the optimal rerouting paths for elephant flows. Experimental results demonstrate that, compared with ECMP (Equal-Cost Multi-Path Routing) and GFF (Global First Fit) traffic scheduling algorithms, the ACO-SDN algorithm reduces the maximum link utilization of the network and effectively improves the network bisection bandwidth.

Full Text

Preamble

Flow Scheduling of Elephant Flows in SDN Data Center Network Based on Ant Colony Algorithm

Li Honghui, Yang Guang, Lu Hailiang, Fu Xueliang, Shen Zhijun
(College of Computer & Information Engineering, Inner Mongolia Agricultural University, Hohhot 010018, China)

Abstract: Traditional methods for scheduling elephant flows in data center networks often cause network congestion and load imbalance. To address these issues, this paper proposes ACO-SDN, a traffic scheduling algorithm for SDN (Software Defined Network) data center networks based on the ant colony algorithm. The elephant flow scheduling problem is formulated as an ILP (Integer

Linear Programming) model with the optimization objective of minimizing maximum link utilization. By redefining the parameters and operations of the ant colony algorithm, the ILP model is solved to obtain optimal rerouting paths for elephant flows. Experimental results demonstrate that compared with ECMP (Equal-Cost Multi-Path Routing) and GFF (Global First Fit) traffic scheduling algorithms, ACO-SDN reduces network maximum link utilization and effectively improves network bisection bandwidth.

Keywords: data center network; software defined network; elephant flow scheduling; ant colony algorithm

0 Introduction

With the rapid development of cloud computing, Internet of Things, and big data technologies, data centers have become the cornerstone of modern computing infrastructure. Communication traffic within data centers is growing exponentially, leading to increasing bandwidth demands on data center networks. Traditional tree-structured data center networks suffer from limited bandwidth and poor scalability, making them unable to meet the bandwidth requirements of internal data center communications. To address this limitation, many novel data center network architectures have been proposed, such as Fat-Tree, Monsoon, BCube, and Helios. Among these, the Fat-Tree topology is relatively simple, provides multiple equivalent paths and higher bisection bandwidth, and facilitates network load balancing.

Currently, data center networks widely utilize the Equal-Cost Multi-Path (ECMP) algorithm for traffic scheduling. When multiple equivalent paths are available to a destination node, ECMP employs static hashing to distribute multiple data flows to these paths, thereby achieving network load balancing and rapid forwarding. However, research shows that data center traffic can be categorized into elephant flows and mouse flows, where elephant flows are defined as those transmitting more than 10% of link bandwidth. Measurements reveal that while 90% of flows are mouse flows, elephant flows carry over 90% of total bytes, meaning the vast majority of data center traffic is concentrated in a small fraction of flows. These two flow types have different performance requirements: elephant flows demand high throughput, while mouse flows require low latency. Studies have demonstrated that ECMP effectively schedules large numbers of mouse flows, but for long-duration elephant flows, it may route multiple such flows onto the same link, causing flow collisions, network congestion, load imbalance, and reduced throughput.

Software Defined Networking (SDN) represents a new programmable network technology that separates the control plane from traditional network devices. Through global centralized control of traffic on the control plane, SDN enables flexible traffic scheduling and resource management optimization, offering new opportunities to solve data center network problems. To address the issues of

network congestion and load imbalance caused by ECMP' s scheduling of elephant flows, this paper proposes ACO-SDN, a dynamic elephant flow scheduling mechanism based on the ant colony algorithm for SDN-enabled Fat-Tree data center networks. First, we establish an Integer Linear Programming (ILP) optimization model for the traffic scheduling problem, with the objective of minimizing maximum link utilization while satisfying all elephant flow scheduling requirements. Then, by redefining the parameters and operations of the ant colony algorithm, we solve the ILP model to obtain an optimized scheduling solution for elephant flows. Based on detected elephant flows and network link utilization, ACO-SDN reroutes elephant flows from a global network perspective. Simulation results show that compared with ECMP and GFF algorithms, ACO-SDN improves data center network bisection bandwidth and reduces maximum link utilization, achieving network load balancing.

1 Related Work

With the rise of SDN technology, many researchers have attempted to use SDN' s centralized control approach to manage data center traffic, achieving real-time flow scheduling through controllers. Hedera proposed a dynamic traffic scheduling mechanism to maximize data center network bisection bandwidth, introducing the Global First Fit (GFF) algorithm for traffic scheduling. For each elephant flow, GFF traverses all possible paths and selects the first path that meets the flow' s bandwidth requirements based on remaining link bandwidth. However, Hedera uses switches to monitor elephant flows, incurring significant additional overhead. To address this issue, Mahout proposed a low-overhead, effective traffic scheduling algorithm that utilizes end hosts for traffic monitoring. When an elephant flow is detected, the host signals the controller, which then calculates a path for the flow.

Tang et al. studied the Dynamic Load Balanced Scheduling (DLBS) problem in SDN data centers. They first established a mathematical optimization model for DLBS with the objective of maximizing network throughput while ensuring dynamic load balancing, then proposed an efficient heuristic algorithm to solve the model and achieve network load balancing within each time slot. Chakraborty and Chen proposed a multipath routing scheme without elephant flow detection to minimize flow completion time. Based on the characteristic that flows exceeding a timeout threshold are removed, this scheme splits elephant flows into multiple mouse flows and routes them across all possible paths, thereby achieving multipath routing in data center networks.

Zhang et al. addressed the elephant flow scheduling problem in SDN data center networks using stable matching theory to model and solve the problem. Their objective was to optimize network performance while avoiding congestion, proposing a new stable matching-based elephant flow scheduling algorithm to achieve high availability and low latency. Zhang et al. also proposed a differ-

entiated scheduling algorithm for SDN-based big data center networks. Based on detected flow types, they used a weighted multipath scheduling algorithm and a blocking island path setup algorithm to schedule mouse flows and elephant flows separately, designing an algorithm to dynamically reschedule flows based on current link utilization to achieve load balancing, with the goal of high throughput, low latency, and load balancing.

To address the flow conflict issues in Hedera's simulated annealing algorithm that failed to consider current network link bandwidth resources, Wang et al. proposed a Simulated Annealing Genetic Algorithm-based Adaptive On-demand (SAGA-AO) traffic scheduling mechanism. This mechanism first screens flows requiring scheduling in the network, then uses SAGA to perform global searches based on current link bandwidth resource conditions to obtain flow scheduling paths, thereby improving average bisection bandwidth in data center networks. To address potential link load imbalance and core switch conflicts in existing traffic scheduling algorithms, Lin et al. proposed a Discrete Particle Swarm Optimization-based Flow Scheduling (DPSOFS) algorithm. This algorithm performs traffic scheduling from a global perspective, effectively and quickly reducing flow conflicts and improving network bisection bandwidth.

Yi et al. proposed a scalable SDN-based data center network segmented routing flow scheduling strategy to address performance bottlenecks caused by low scalability of traffic routing mechanisms in large-scale SDN data center networks. Using edge switches to detect elephant flows and to meet different QoS guarantees and network scalability requirements, they proposed an online first-fit algorithm. Simulation results showed that this mechanism reduced total controller overhead while improving network throughput. Jin and Shu addressed the problem of network congestion and load imbalance caused by elephant flows carrying large amounts of data in data center networks, proposing an SDN-based elephant flow load balancing mechanism (EFLB). This mechanism monitors the network in a round-robin fashion. When load exceeds a threshold, the controller splits detected elephant flows into multiple mouse flows and calculates the next-hop switch with minimum load to ensure load balancing, thereby improving network throughput, reducing latency, and better achieving network load balancing.

---

2 Ant Colony Traffic Scheduling Algorithm

This chapter first introduces the ant colony algorithm-based traffic scheduling mechanism ACO-SDN, including elephant flow detection and network state collection methods. It then establishes the optimization mathematical model ILP for the traffic scheduling problem, and finally presents the ant colony algorithm-based method for solving the ILP model.

2.1 Elephant Flow Traffic Scheduling Mechanism ACO-SDN

The flow scheduling mechanism proposed in this paper is illustrated in Figure 1 [Figure 1: see original paper]. The specific traffic scheduling process is as follows:

a) When the data center network receives data flows from hosts, it first schedules them using the ECMP algorithm. Simultaneously, sFlow agents in the switches collect network state information, including link usage and data flow information, and transmit the collected information to the sFlow collector in the controller every 2 seconds.

b) Based on the received information, the sFlow collector calculates link utilization and identifies elephant flows. If a flow' s bandwidth exceeds 10% of link capacity, it is marked as an elephant flow.

c) The sFlow collector passes the elephant flow information and link utilization to the Floodlight controller.

d) The controller determines whether the link utilization on the path of the identified elephant flow exceeds the threshold of 60%. If yes, proceed to step e); otherwise, return to step a).

e) Based on the current link usage state, the controller invokes the ant colony algorithm to solve the traffic scheduling mathematical model and calculates an optimal path for the elephant flow. This path is converted into flow table entries and installed in the switches.

f) The switches reroute elephant flows on congested links according to the new flow table entries, then return to step a).

2.2 Traffic Scheduling Problem Mathematical Model

The data center network topology is represented as a graph $G(V, L)$, where V denotes the set of all nodes in the network and L denotes the set of all links. For node v_i, L_i^{in} represents the set of incoming links for all data flows, and L_i^{out} represents the set of outgoing links. The capacity of link l is C_l, and its utilization is u_l. Let F be the set of elephant flows requiring scheduling in the current network, with each flow $f \in F$ having bandwidth b_f, source point s_f, and destination point t_f. The variable x_{fl} is introduced to indicate whether elephant flow f is routed through link l.

Link utilization is defined as the ratio of the sum of bandwidths of all elephant flows traversing that link to the link capacity, as shown in Equation (1):

$$u_l = \frac{\sum_{f \in F} b_f x_{fl}}{C_l}$$

Maximum link utilization u_{max} is the maximum value among all link utilizations in the network:

$$u_{max} = \max_{l \in L} u_l$$

The mathematical model for the traffic scheduling problem is formulated as follows:

$$\min \quad u_{max}$$

Subject to:

$$\sum_{f \in F} b_f x_{fl} \leq C_l \quad \forall l \in L \tag{4}$$

$$\sum_{l \in L_i^{in}} x_{fl} - \sum_{l \in L_i^{out}} x_{fl} = \begin{cases} -1, & \text{if } i = s_f \\ 1, & \text{if } i = t_f \\ 0, & \text{otherwise} \end{cases} \quad \forall f \in F, \forall v_i \in V \tag{5}$$

$$x_{fl} \in \{0, 1\} \quad \forall f \in F, \forall l \in L \tag{6}$$

Equation (3) represents the optimization objective of minimizing the maximum link utilization. Equations (4) through (6) are constraints that must be satisfied during traffic scheduling. Equation (4) ensures that the total bandwidth of data flows carried by link l does not exceed the link capacity. Equation (5) is the flow conservation constraint, ensuring that for each intermediate node, the incoming flow equals the outgoing flow. Equation (6) defines the variable domain.

This mathematical model belongs to the Integer Linear Programming (ILP) class. To obtain the routing schedule for elephant flows, this paper employs the ant colony optimization algorithm to solve the model.

2.3 Ant Colony Optimization Algorithm

Both ant colony algorithms and genetic algorithms are powerful tools for solving the ILP model. The difference lies in that ant colony algorithms utilize pheromone accumulation and updating to obtain optimal solutions, whereas genetic algorithms have strong global solution space search capabilities but suffer from redundant iterations due to lack of feedback information, resulting in lower solution efficiency.

The ant colony algorithm, first proposed by Marco Dorigo, is an approximate optimization algorithm for finding optimal paths, inspired by ant colonies' behavior in discovering shortest paths during foraging. Ants release pheromones on their traversed paths for information communication. Other ants in the colony sense these pheromones and tend to follow paths with higher pheromone concentrations, while each passing ant further releases pheromones, creating a

positive feedback mechanism. Over time, the entire ant colony converges on the optimal path to the food source.

The basic idea of using the ant colony algorithm to solve the ILP model is to represent feasible routing solutions for elephant flows as ant foraging paths, with all ant paths constituting the solution space of optional routes. Ants on shorter paths release more pheromones, and over time, pheromone concentrations on shorter paths gradually increase, attracting more ants. Eventually, the entire ant colony concentrates on the best path through positive feedback, corresponding to the optimal solution for elephant flow routing. The specific algorithm steps are as follows:

a) **Parameter Initialization**: Parameters are divided into network state parameters, elephant flow parameters, and ant colony algorithm parameters. Network state parameters include node set V, link set L, link capacity C_l, and link utilization u_l. Elephant flow parameters include bandwidth b_f, source s_f, destination t_f, and flow duration for each flow f requiring scheduling. Ant colony algorithm parameters include number of ants m, initial ant positions, total pheromone amount Q, pheromone heuristic factor α, expectation heuristic factor β, pheromone volatilization coefficient ρ, initial pheromone $\tau_{ij}(0)$ on each link, maximum number of nodes N_{max} an ant can traverse, and algorithm iteration count I. The tabu link table TB_k for ant k stores traversed links.

b) **Path Construction**: $\tau_{ij}(t)$ represents the pheromone concentration on the path from node i to j at time t, and $\eta_{ij}(t)$ represents the heuristic information, defined as the inverse of link utilization: $\eta_{ij}(t) = \frac{1}{u_{ij}}$. During traversal of all feasible paths, each ant k selects the next node j based on $p_{ij}^k(t)$ and adds the traversed link to its tabu link table TB_k. The selection probability is:

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{n \in V_k} [\tau_{in}(t)]^\alpha [\eta_{in}(t)]^\beta}, & j \in V_k \\ 0, & \text{otherwise} \end{cases}$$

where V_k represents the set of nodes that ant k is allowed to visit next.

c) **Iteration Process**: Repeat step b) until all ants have traversed N_{max} nodes, then terminate the current iteration and increment the iteration count.

d) **Pheromone Update**: For ants k that successfully reach the destination, update the pheromone concentration on their paths using:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t)$$

where $\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^k(t)$. The pheromone increment $\Delta\tau_{ij}^k(t)$ for ant k is

$\frac{Q}{L_k}$ if the ant traversed link (i, j), and 0 otherwise, with L_k being the path length of ant k in the current iteration.

e) **Termination**: After completing the specified maximum iteration count I, stop path searching and select the optimal path from all ants' tabu tables according to the optimization objective.

To prevent the ant colony algorithm from falling into local optima, this paper adopts two measures: (a) differential initialization of pheromone concentrations on network links by setting initial pheromone on K-shortest paths between source and destination to $\frac{Q}{\text{path length}}$ while setting others to 0, and (b) employing the roulette wheel algorithm to enhance global search capability and further improve solution quality.

---

3 Experimental Results and Analysis

To evaluate the performance of the ACO-SDN optimization algorithm, we built a Fat-Tree data center network using the Floodlight controller and Mininet simulation platform, employing average bisection bandwidth and maximum link utilization as performance metrics. Bisection bandwidth refers to the total bandwidth of the minimum number of links that must be removed to divide all hosts in a network into two equal parts; larger bisection bandwidth indicates stronger overall network transmission performance. Through comparison with the GFF algorithm and the widely used ECMP algorithm, experimental results demonstrate that ACO-SDN exhibits advantages in average bisection bandwidth and maximum link utilization on the most congested links.

3.1 Experimental Setup

1) Network Topology
On the Mininet simulation platform, we constructed a $k = 4$ Fat-Tree data center network architecture, as shown in Figure 2 [Figure 2: see original paper], where all switches are OpenFlow switches. All links are configured with 100 Mb/s bandwidth.

2) Network Load
Due to privacy and security concerns, no publicly available data center network load data exists. Recent studies on data center traffic characteristics indicate that traffic matrices vary dramatically over time and space. To evaluate the proposed elephant flow scheduling mechanism ACO-SDN, we determined communication patterns based on literature [3,13,18,19]. The Iperf tool generates data traffic with flow lengths following an exponential distribution, flow arrival times following a Poisson distribution, and destination hosts determined by one of three communication patterns:

a) **Stride(i) Pattern**: Host with index x sends data to host with index

$(x+i) \bmod n$, where n is the number of hosts in the network.

b) **Staggered(p_{Edge}, p_{Pod}) Pattern**: Each host sends data to hosts in the same edge switch with probability p_{Edge}, to hosts in the same pod with probability p_{Pod}, and to hosts in other pods with probability $1 - p_{Edge} - p_{Pod}$.

c) **Random Pattern**: Each host randomly sends data to other hosts in the network with equal probability.

3) Algorithm Parameter Settings

As described in Section 2.3, ant colony optimization parameters are divided into network parameters, elephant flow parameters, and ant colony algorithm parameters. During simulation experiments, these parameters are set as follows:

a) Network Parameters

The node set V includes all nodes in the Fat-Tree data center network shown in Figure 2 [Figure 2: see original paper], comprising both switches and servers. The link set L includes all links in this network, with each link capacity set to 100 Mb/s. Link utilization u_l is calculated based on current link load using Equation (1).

b) Elephant Flow Parameters

Source s_f and destination t_f of elephant flow f are server nodes, set according to the communication pattern when generating flows. Elephant flows are defined as those exceeding 10% of link capacity, so bandwidth b_f is set as a random value between 10 Mb/s and 20 Mb/s. Flow length is determined when generating the elephant flow.

c) Ant Colony Algorithm Parameters

Algorithm parameter settings directly affect ant colony optimization performance. Based on literature [27,28], traffic scheduling requirements, and extensive simulation results, main parameters are set as shown in Table 1 .

Table 1: Main Parameter Values for Ant Colony Algorithm

Parameter	Symbol	Value Range	Setting
Number of Ants	m	$[0.6n, 0.9n]$	0.8n
Pheromone Heuristic Factor	α	$[1, 5]$	2
Expectation Heuristic Factor	β	$[1, 5]$	3
Pheromone Volatilization Coefficient	ρ	$[0.5, 0.8]$	0.6
Total Pheromone Amount	Q	-	100
Initial Ant Position	-	-	Source of flow to be scheduled
Ant Tabu Table	TB_k	-	Initially empty
Iteration Count	I	$[50, 100]$	75
Maximum Nodes Traversed	N_{max}	-	10

The initial ant position is set to the source of the elephant flow to be scheduled, and the ant tabu table is initialized as empty. Through multiple experimental tests, algorithm performance stabilizes when iteration count is between 50 and 100, so we set I to the intermediate value of 75. During simulation, path length between two nodes is limited to 11 hops. In a $k = 4$ Fat-Tree network, paths exceeding 11 hops are highly unlikely to be optimal, so the maximum number of nodes traversed (switches) is set to 10. Initial pheromone $\tau_{ij}(0)$ on each link is set as described in Section 2.3.

3.2 Average Bisection Bandwidth Comparison

To compare the average bisection bandwidth of ECMP, GFF, and ACO-SDN algorithms, we conducted multiple experiments using the three communication patterns described above. Each experiment lasted approximately 60 seconds, with bisection bandwidth measured from the middle portion.

1) Stride Communication Pattern
Under the Stride(i) pattern, we tested three cases: $i = 4$, $i = 6$, and $i = 8$. Figure 3 [Figure 3: see original paper] shows the average bisection bandwidth comparison for ECMP, GFF, and ACO-SDN under Stride(4), Stride(6), and Stride(8) modes. The results demonstrate that ACO-SDN achieves higher average bisection bandwidth than both ECMP and GFF in all three modes. Specifically, ACO-SDN improves average bisection bandwidth by 19%-26% compared to ECMP and by 8%-17% compared to GFF.

2) Staggered Communication Pattern
Under the Staggered pattern, we tested Staggered(0, 0.2) and Staggered(0, 0.4) modes, with two experiment groups for each. Figure 4 [Figure 4: see original paper] compares the average bisection bandwidth of the three algorithms. ACO-SDN outperforms both ECMP and GFF in these two modes, improving average bisection bandwidth by 8%-21% compared to ECMP and by 5%-15% compared to GFF.

3) Random Communication Pattern
Under the Random pattern, we conducted three experiment groups: Random1, Random2, and Random3. Figure 5 [Figure 5: see original paper] shows the comparison results. ACO-SDN achieves superior average bisection bandwidth compared to GFF and ECMP, with improvements of approximately 23% over ECMP and 14% over GFF.

In summary, across all three communication patterns, ACO-SDN' s average bisection bandwidth exceeds that of ECMP and GFF. ECMP performs worst because it randomly schedules elephant flows onto a shortest path without considering current link state, increasing flow collisions. GFF reduces collisions by selecting the first satisfactory path based on current link utilization, thus improving average bisection bandwidth over ECMP. ACO-SDN calculates globally optimal paths based on current link utilization, minimizing elephant flow conflicts and maximizing network bisection bandwidth.

3.3 Maximum Link Utilization Comparison

To further validate algorithm performance, we compared ACO-SDN, ECMP, and GFF from the perspective of maximum link utilization, which reflects bandwidth usage across the network. Balanced network load should result in relatively uniform link utilization across paths from source to destination, fully utilizing multipath advantages to reduce flow conflicts and improve throughput.

We selected the Staggered(0, 0.3) pattern with hosts in the first pod as data sources sending flows to hosts in other pods. Flow bandwidth was fixed at 20 Mb/s with 60-second duration. This experiment intentionally creates network load imbalance and high link load to better evaluate maximum link utilization. Multiple experiments were conducted using ECMP, GFF, and ACO-SDN until stable results were obtained.

Figure 6 [Figure 6: see original paper] shows the cumulative distribution function (CDF) of maximum link utilization (MLU) for the three algorithms. The x-axis represents MLU, and the y-axis represents the CDF. The results show that ACO-SDN reduces maximum link utilization by approximately 5% compared to GFF and by about 8% compared to ECMP. ACO-SDN' s MLU concentrates between 60%-70%, while ECMP' s MLU concentrates between 70%-80%. This improvement occurs because ACO-SDN continuously monitors links on elephant flow paths. When link utilization exceeds the threshold, ACO-SDN recalculates a new path based on current network state to reroute elephant flows, reducing maximum link utilization and conflicts. ECMP, as a static algorithm, randomly routes elephant flows to shortest paths without considering current link state, increasing collision probability and maximum link utilization, resulting in the poorest performance. GFF considers link utilization when routing elephant flows, performing better than ECMP, but routes flows to the first satisfactory path in arrival order, placing its maximum link utilization between ACO-SDN and ECMP.

---

4 Conclusion

To address the problems of elephant flow collisions, network congestion, and performance degradation caused by traditional traffic scheduling methods, this paper proposes an SDN-based Fat-Tree data center network elephant flow scheduling mechanism. We first formulate the elephant flow scheduling problem as an Integer Linear Programming (ILP) optimization model and solve it using the ant colony optimization algorithm to obtain near-optimal scheduling solutions. Simulation experiments using the Floodlight controller and Mininet platform demonstrate that the proposed ACO-SDN mechanism outperforms ECMP and GFF algorithms. Under Stride, Staggered, and Random communication patterns, ACO-SDN significantly improves network average bisection bandwidth while reducing maximum link utilization.

References

[1] Cai Yueping, Wang Changping. Software defined data center network with hybrid routing [J]. Journal on Communications, 2016, 37 (4): 44-52.

[2] Curtis A R, Kim W, Yalagandula P. Mahout: low-overhead datacenter traffic management using end-host-based elephant detection [C]// Proc of IEEE INFOCOM. 2011: 1629-1637.

[3] Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture [C]// Proc of ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. 2008: 63-74.

[4] Greenberg A, Lahiri P, Maltz D A, et al. Towards a next generation data center architecture: scalability and commoditization [C]// Proc of ACM Workshop on Programmable Routers for Extensible Services of Tomorrow. [S. l.] : ACM Press, 2008: 57-62.

[5] Guo Chuanxiong, Lu Guohan, Li Dan, et al. BCube: a high performance, server-centric network architecture for modular data centers [J]. ACM SIGCOMM Computer Communication Review, 2009, 39 (4): 63-74.

[6] Farrington N, Porter G, Radhakrishnan S et al. Helios: a hybrid electrical//optical switch architecture for modular data centers [J]. ACM SIGCOMM Computer Communication Review, 2010, 40 (4): 339-350.

[7] Escudero-Sahuquillo J, Garcia P J, Quiles F J, et al. A new proposal to deal with congestion in InfiniBand-based fat-trees [J]. Journal of Parallel & Distributed Computing, 2014, 74 (1): 1802-1819.

[8] Hopps C. Analysis of an equal-cost multi-path algorithm [S]. RFC 2992, 2000.

[9] Kandula S, Sengupta S, Greenberg A, et al. The nature of data center traffic: measurements & analysis [C]// Proc of ACM SIGCOMM Conference on Internet Measurement. 2009: 202-208.

[10] Benson T, Akella A, Maltz D A. Network traffic characteristics of data centers in the wild [C]// Proc of the 10th ACM SIGCOMM Conference on Internet Measurement. 2010: 267-280.

[11] Li Dan, Xu Mingwei, Zhao Hongze, et al. Building mega data center from heterogeneous containers [C]// Proc of IEEE International Conference on Network Protocols. [S. l.] : IEEE Press, 2011: 256-265.

[12] Mckeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks [J]. ACM SIGCOMM Computer Communication Review, 2008, 38 (2): 69-74.

[13] Al-Fares M, Radhakrishnan S, Raghavan B, et al. Hedera: dynamic flow scheduling for data center networks [C]// Proc of Usenix Symposium on Networked Systems Design and Implementation. 2010: 281-296.

[14] Tang Feilong, Yang L T, Tang Can, et al. A dynamical and load-balanced flow scheduling approach for big data centers in clouds [J]. IEEE Trans on Cloud Computing, 2016, 99 (1) , 1-14.

[15] Chakraborty Suchandra, Chen Chien. A low-latency multipath routing without elephant flow detection for data centers [C]// Proc of IEEE International Conference on High Performance Switching and Routing. Yokohama: IEEE Press, 2016: 49-54.

[16] Zhang Yuxiang, Cui Lin, Zhang Yuan. A stable matching based elephant flow scheduling algorithm in data center networks [J]. Computer Networks, 2017, 120 (2017): 186-197.

[17] Zhang Heteng, Tang Feilong, Barolli L. Efficient flow detection and scheduling for SDN-based big data centers [J/OL]. Journal of Ambient Intelligence & Humanized Computing, 2018, https://doi. org/10. 1007/s12652-018-0783-6.

[18] Wang Wentao, Zheng Fang, Wang Lingxia, et al. Design and implementation of flow scheduling mechanism based on SDN for data center network [J]. Journal of South-Central University for Nationalities: Nat. Sci. Edition, 2016, 35 (3): 135-140.

[19] Lin Zhihua, Gao Wen, Wu Chunming, Li Yongyan. Data center network flow scheduling based on DPSO algorithm [J] , Acta Electronica Sinica, 2016, 44 (9): 2197-2202.

[20] Yi Peng, Liu Hong, Hu Yuxiang. A scalable traffic scheduling policy for software defined data center network [J]. Journal of Electronics & Information Technology, 2017, 39 (4): 825-831.

[21] Jin Ling, Shu Yongan. Research on load balancing of elephant flow based on SDN in data center network [J/OL]. Application Research of Computers, 2019 (1): 1-5. [2018-05-30]. http://kns. cnki. net/kcms/detail/51. 1196. TP. 20180208. 1714. 094. html.

[22] Lantz B, Heller B, Mckeown N. A network in a laptop: rapid prototyping for software-defined networks [C]// Proc of ACM Workshop on Hot Topics in Networks. 2010: 1-6.

[23] Kang Fenglan, Li Kangshun. A comparison study of GA and ACA on TSP [J]. Computer Systems & Applications, 2008, 17 (10): 60-63.

[24] Duan Haipin, Zhang Xiangyin, Xu Chunfang. Bio-inspired computing [M]. Beijing: Science Press, 2011.

[25] Ma Zhen. Research on the improvement of ant colony algorithm and its application in TSP [D]. Qingdao: Qingdao University of Technology, 2016.

[26] Greenberg A, Hamilton J R, Jain N, et al. VL2: a scalable and flexible data center network [J]. Communications of the Acm, 2009, 54 (4): 95-104.

[27] Zhan Shichang, Xu Jie, Wu Jun. The optimal selection on the parameters of the ant colony algorithm [J]. Bulletin of Science and Technology, 2003, 19 (5): 381-386.

[28] Xu Hongmei, Chen Yibao, Liu Jiaguang, et al. The research on the parameters of the ant colony algorithm [J]. Journal of Shandong University of Technology: Natural Science Edition, 2008, 22 (1): 7-11.

Note: Figure translations are in progress. See original paper for figures.