Frontiers of Information Technology & Electronic Engineering
www.zju.edu.cn/jzus; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

An OpenFlow-based performance-oriented multipath forwarding scheme in datacenters*

Bo LIU†1, Ming CHEN†1, Bo XU†‡1, Hui HU1, Chao HU1, Qing-yun ZUO2, Chang-you XING1

(1College of Command Information Systems, PLA University of Science and Technology, Nanjing 210007, China)

(2PLA Academy of National Defense Information, Wuhan 430020, China)

†E-mail: lbo.xidian@163.com; mingchennj@163.com; xubo820@163.com

Received Mar. 7, 2016; Revision accepted June 23, 2016; Crosschecked June 28, 2016

Abstract: Although dense interconnection datacenter networks (DCNs) (e.g., FatTree) provide multiple paths and high bisection bandwidth for each server pair, the widely used single-path Transmission Control Protocol (TCP) and equal-cost multipath (ECMP) transport protocols cannot achieve high resource utilization due to poor resource excavation and allocation. In this paper, we present LESSOR, a performance-oriented multipath forwarding scheme to improve DCNs' resource utilization. By adopting an OpenFlow-based centralized control mechanism, LESSOR computes near-optimal transmission path and bandwidth provision for each flow according to the global network view while maintaining nearly real-time network view with the performance-oriented flow observing mechanism. Deployments and comprehensive simulations show that LESSOR can efficiently improve the network throughput, which is higher than ECMP by 4.9%–38.3% under different loads. LESSOR also provides 2%–27.7% improvement of throughput compared with Hedera. Besides, LESSOR decreases the average flow completion time significantly.

Key words: Datacenter network, Traffic engineering, OpenFlow, Multipath transmission
http://dx.doi.org/10.1631/FITEE.1601059 **CLC number:** TP393

1 Introduction

Nowadays, dense interconnection datacenter networks (DCNs) (Qi *et al.*, 2014), e.g., FatTree (Al-Fares *et al.*, 2008) and VL2 (Greenberg *et al.*, 2011), have been deployed by cloud computing providers as well as enterprises to provide large bisection bandwidth for the ever-increasing array of applications, ranging from financial services to big-data analytics. However, the traditional TCP/IP (Transmission Control Protocol/Internet Protocol) architecture applies distributed control mechanisms, which makes some links congested due to oversubscription but other links having light load. The average bandwidth utilization under the single-path TCP transmission scheme is only 30%–40%. Therefore, there is an intense demand for improving resource utilization in DCNs.

Several multipath transport schemes have been proposed to improve bandwidth utilization in DCNs, such as multipath-TCP (MPTCP) (Ford *et al.*, 2013) and equal-cost multipath (ECMP) (Hopps, 2000), which have obtained better bandwidth utilization than TCP. More specifically, MPTCP (Raiciu *et al.*, 2011b) increases the lowest link utilization to more than 50%, while ECMP (Chiesa *et al.*, 2014) improves the average network throughput to 80%, but

‡ Corresponding author

* Project supported by the National Basic Research Program (973) of China (No. 2012CB315806), the National Natural Science Foundation of China (Nos. 61103225 and 61379149), the Jiangsu Provincial Natural Science Foundation (No. BK20140070), and the Jiangsu Future Networks Innovation Institute Prospective Research Project on Future Networks, China (No. BY2013095-1-06)

ORCID: Bo LIU, http://orcid.org/0000-0003-2789-7201

the performance of these schemes is still far from optimal for short of global optimization. Hedera (Al-Fares *et al.*, 2010) and MicroTE (Benson *et al.*, 2011) introduce OpenFlow (McKeown *et al.*, 2008) in DCNs to improve the network throughput, demonstrating that OpenFlow can enhance the instant management and provide fine-grained control on network resource allocation owing to the centralized control mechanism. However, due to poor resource excavation and allocation, they still have a wide performance gap compared with the optimal. Packet- or flowlet-level schemes suffer from the packet disordering problem and are hard to implement. So, we prefer a flow-level and easily deployable scheme. OpenFlow is easy to deploy in DCNs (Jain *et al.*, 2013), which also supports incremental deployment. So, we use OpenFlow to address traffic engineering issues in this study.

Based on OpenFlow's centralized control mechanism, LESSOR seamlessly integrates fine-grained resource excavation and precise resource allocation mechanism to achieve high resource utilization in DCNs. Unlike other solutions (Al-Fares *et al.*, 2010; Raiciu *et al.*, 2011a; Li *et al.*, 2012; Chiesa *et al.*, 2014), LESSOR adopts a performance-oriented multipath transmission mechanism (PMPT) rather than ECMP or MPTCP, since the random load balancing solutions seriously impair network throughput while it is too arduous to deploy MPTCP immediately in DCNs. In brief, PMPT aims to allocate enough bandwidth for each flow to meet its demand and accommodate more concurrent flows by maintaining good load balancing as well. Moreover, LESSOR employs a performance-oriented resource excavation (PRE) mechanism to obtain real-time network views. In LESSOR, end servers just inject the traffic into network, and the controller implements LESSOR to execute resource allocation. Therefore, LESSOR does not request end servers to participate in and complies with OpenFlow standards strictly; thus, it is readily implementable. The main contributions of this study are as follows:

1. We design and implement LESSOR, an OpenFlow-based performance-oriented multipath forwarding scheme, for DCNs. LESSOR is immediately deployed, efficient, and precise in allocating network resources.

2. We propose PRE and PMPT mechanisms, where PRE provides more fine-grained network resource excavation while PMPT achieves better performance in network resource usage compared with existing schemes.

3. We test the performance of LESSOR with real datacenter traffic patterns in a Mininet emulator and implement it on a real testbed. Experimental results show that LESSOR improves resource utilization compared to TCP and ECMP as well as other existing schemes.

2 Motivation

In this section, we introduce some significant observations in maximizing resource utilization and decreasing the average flow completion time (AFCT) in high-interconnection network topologies. As shown in Fig. 1, each link's bandwidth is 10 Mb/s. Four flows transmitting from h1 to h2 arrive at the same time, and the flow sizes are listed in Table 1. Three different routing schemes are used in the example, which are TCP, ECMP, and the optimal routing (OR) scheme. The routings are shown in Figs. 1a–1c, and the corresponding optimal scheduling schemes (short job first, SJF) are shown in Figs. 1d–1f.

Table 1 Flow sizes used in the example

Flow	f1	f2	f3	f4
Size (Mb)	2	4	8	6

Observation 1 The multipath transport scheme is more efficient than TCP in improving throughput and decreasing AFCT. Fig. 1 shows that TCP has the lowest throughput in the three schemes. Specifically, its throughput is only half of the optimal. Meanwhile, it can be concluded that the multipath transport scheme can efficiently decrease FCT since the total FCT is 2 s in TCP while it is 1 s in OR.

Observation 2 Precise global optimization is more efficient than distributed control in bandwidth allocation. In our example, ECMP achieves only 83.3% of OR in throughput since ECMP is short of excellent resource excavation and allocation mechanisms. Therefore, a precise global bandwidth allocation mechanism is advisable to improve network resource utilization.

Observation 3 It is essential to adopt the performance-oriented resource allocation mechanism.

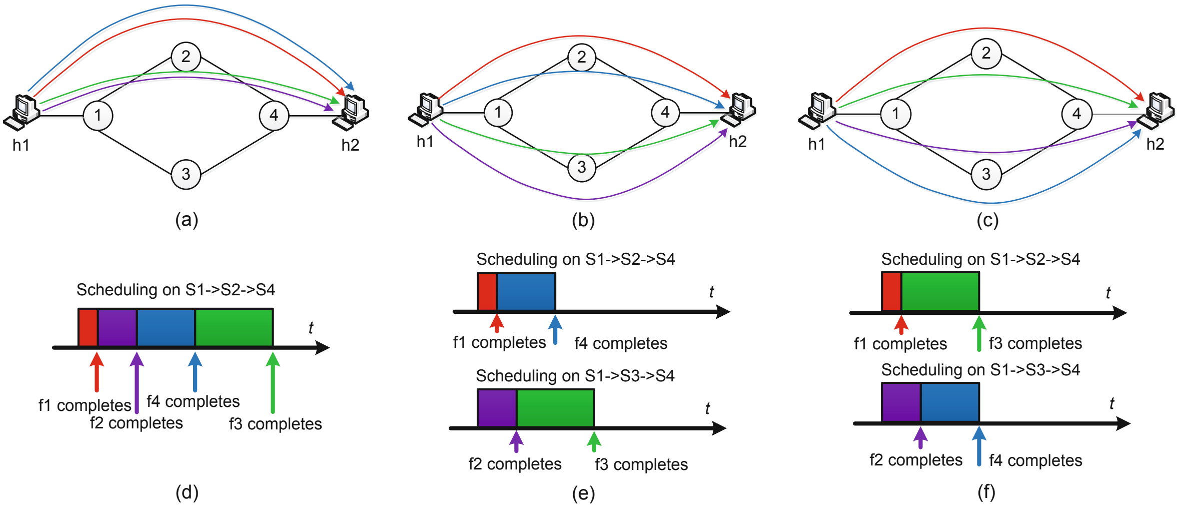

Fig. 1 An explanatory example: (a) routing with TCP; (b) routing with ECMP; (c) optimal routing scheme; (d)–(f) optimal scheduling schemes for (a)–(c), respectively

For example, it is preferable to improve load balancing to maximize bandwidth utilization and schedule short flow first to decrease AFCT.

Actually, the performance of the above resource allocation schemes is heavily dependent on the correct network resource information, because erroneous network information would cause an improper resource allocation, thereby impairing network performance (Yu *et al.*, 2013).

In conclusion, to maximize the network resource utilization, a centralized control scheme is advisable and an excellent resource excavation mechanism should be seamlessly integrated. Moreover, a performance-oriented resource allocation mechanism is essential to make optimal use of the network resource. In our example, the highest throughput and the minimum AFCT can be achieved by an optimal multipath transport scheme with its scheduling mechanism shown in Figs. 1c and 1f, respectively. LESSOR is thus designed to accomplish these targets.

3 Design overview

LESSOR uses an OpenFlow-based, centralized, cooperative method, which is coherent with many recent centralized datacenter designs (Al-Fares *et al.*, 2010; Benson *et al.*, 2011). It aims at maximizing network resource utilization, where the network resource includes both the transmission path and link bandwidth. Specifically, for each flow, LESSOR determines which path carries it, when to start it, and how much bandwidth should be allocated to it. Therefore, LESSOR first needs to maintain a network view approximating to real-time network state as far as possible. For example, LESSOR should be aware of the network topology (Le *et al.*, 2014) and the workload in each link, and get more details of each flow such as the size and sending rate. Furthermore, LESSOR should execute an efficient and precise resource allocation mechanism to maximize network efficiency.

To make LESSOR easily deployable, all control functions are implemented in the controller. OpenFlow switches merely forward packets in line with the flow table installed by the controller and fulfill the job of traffic statistics relying on their inside functions. There are two main modules in LESSOR, PRE and PMPT. PRE aims to excavate all the available resources in DCNs and PMPT targets on executing near-optimal resource allocation. More details of LESSOR modules are described below.

3.1 PRE module

PRE excavates network resource by using the comprehensive information query system of the OpenFlow technology. Specifically, PRE can send symmetric messages such as OFP_ECHO_REQUEST to check whether an OpenFlow switch still works, while sending OFP_PORT_STATUS to check the state of a given switch port and find the failure link. Moreover, by sending OFP_STATS_REQUEST, PRE can effortlessly obtain the transmission rate of each flow and bandwidth usage amount of each link. However, it is unable to perceive network resource in real time, because the controller cannot obtain the above information in one round-trip time (RTT). PRE adopts a dynamic polling period, which changes according to the CPU utilization of the controller, for obtaining more precise resource information. Generally speaking, PRE prefers a shorter polling period when the CPU utilization of the controller is low (controller is idle) without adding much more overhead on the controller, so other functions running in the controller will be unaffected. Meanwhile, PRE leverages a short-term and partial predictability of the traffic matrix. For workload awareness, PRE assumes that the information about each flow can be derived from the upper layer applications or using the state-of-the-art prediction techniques (Wilson *et al.*, 2011; Peng *et al.*, 2014), and it mandates that flows less than 100 KB are considered as short flows. In LESSOR, end servers label long and short flows, and we realize this according to Benson *et al.* (2011).

3.2 PMPT module

PMPT precisely computes the near-optimal path and bandwidth reservation for each flow and maximizes resource utilization. MicroTE (Benson *et al.*, 2011) shows that more than 80% of flows are short flows and they represent a very small fraction (<5%) of the overall traffic, while less than 5% of long flow contributes more than 90% of the total bytes. To achieve scalability, PMPT orchestrates only long flows, while short flows are treated as background traffic to decrease the overhead of the controller. Background traffic is transmitted with the preinstalled flow tables installed by the controller, which will contribute to decreasing the short flow completion time (SFCT) by overleaping the extra RTT in installing flow tables. Meanwhile, PMPT employs the well-known minimum remaining time first (MRTF) (Hong *et al.*, 2012) principle to induce AFCT in resource allocation.

The framework of LESSOR is illustrated in Fig. 2. Both PRE and PMPT realize their functions based on controller programs such as POX (http://www.noxrepo.org/pox/about-pox/), while the controller communicates with OpenFlow switches abiding by the OpenFlow protocol (http://www.openflowswitch.org/documents/openflow-spec-v1.0.0.pdf). PRE and PMPT work independently but they cooperatively maintain the network view. The network view in our scheme is composed of a series of data structures created and updated by the controller, PRE, and PMPT. Specifically, the controller concentrates on monitoring the changes of the network hardware entities such as the connecting state of the switches, switch ports, and end servers, while PRE observes mainly network traffic, link bandwidth, and the information of transmission paths. PMPT will modify the related data after computing the transmission path and bandwidth reservation for flows. Meanwhile, PRE and PMPT implement a closed-loop control method, which can work well even though single query information is inaccurate. More details are given in the next section.

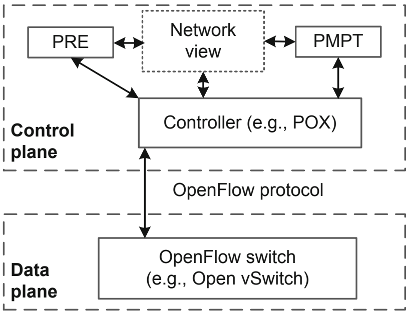

Fig. 2 Framework of LESSOR

4 Details of LESSOR

In this section, we first present the mathematical model of LESSOR (Section 4.1). To solve this problem, we compute the network information with the PRE algorithm (Section 4.2), and next present the PMPT algorithm to near-optimally compute the routing and corresponding bandwidth reservation for each flow (Section 4.3).

4.1 Maximizing resource utilization

A directed graph $G = (V, E)$ is used to describe the network model, with the assumption that the network is composed of a set of nodes (V) and a set of directed links (E). Moreover, let $f(e)$ and $c(e)$

represent the current traffic and the link capacity of link e ($e \in E$), respectively, and $\mathrm{NH}(u, d)$ the set of next hop nodes from u to d.

Given two servers s and d, any path $< s, u_0, u_1, \ldots, u_k, d >$ from the source server s to the destination server d will be termed a reachable path if $(u_{j-1}, u_j) \in E$ and $u_j = \mathrm{NH}(u_{j-1}, d)$ for $j = 1, 2, \ldots, k$, and the set of reachable paths between s and d is denoted by P_{sd}. Besides, let W_{sd} represent the bisection bandwidth between s and d.

The target of LESSOR is to improve network resource utilization, i.e., to improve the utilization parameter λ of a given bisection bandwidth. We introduce the following optimization problem to describe the goal of traffic decomposition:

$$\text{maximize } \lambda$$
$$\text{s.t.} \begin{cases} x_{sd}(p) + f(e) \le c(e), \ \forall e \in E, \ \forall p \in P_{sd}, & (1) \\ \displaystyle\sum_{p \in P_{sd}} x_{sd}(p) \le \lambda W_{sd}, \ \forall s \in V, \ \forall d \in V, & (2) \\ x_{sd}(p) \ge 0, \qquad\qquad \forall p, & (3) \end{cases}$$

where $x_{sd}(p)$ denotes the traffic constituent on path p from s to d.

Constraint (1) indicates that the sum of the existing traffic $f(e)$ and all new arriving traffic $x_{sd}(p)$ on link e must be less than the capacity of the link, $c(e)$. Constraint (2) means that the total arriving traffic cannot exceed the product of the maximum utilization parameter λ and the bisection bandwidth W_{sd}. Constraint (3) ensures that the flow on any path is non-negative.

This model is a maximum multicommodity flow problem and thus is NP-hard. To solve this problem, LESSOR needs to compute $f(e)$ and W_{sd} dynamically to reflect the real-time network workloads, for which it uses PRE to compute $f(e)$ and W_{sd} with high quality. After that, LESSOR uses PMPT to compute near-optimal bandwidth allocation for each flow and the value of λ.

4.2 Performance-oriented resource excavation

In the OpenFlow network, the controller can obtain the transmission rate of each flow and the available bandwidth of each link by sending related request messages to OpenFlow switches, and the controller can obtain a more accurate statistic by virtue of a smaller query period. However, the controller also needs to analyze the packet-in messages, compute the path for each flow, and execute other functions. Meanwhile, these functions will add extra control overhead to the centralized controller, which is always reflected by the CPU and memory usage of the controller. Therefore, to guarantee that the controller is not overloaded, the query job should not be executed frequently. Existing schemes (Al-Fares *et al.*, 2010; Kabbani *et al.*, 2014) always use a roughly static polling period to decrease the overhead of the controller by sacrificing the accuracy of the statistic. However, the controller is not always busy, so the schemes with static polling time cannot make full use of the power of the controller to seek high performance. Thus, we propose to use dynamic polling time to obtain more accurate flow statistics. Benson *et al.* (2011) and Curtis *et al.* (2011b) required the controller to query all OpenFlow switches for the statistics of flows and links. This is unnecessary because the residuary bandwidth of links between aggregation switches and spine switches can be calculated after obtaining all flows' transmitting rates from top of rack (ToR) switches. So, PRE queries only ToR switches to obtain flow information for decreasing the overhead of the controller.

An upper utilization threshold of CPU should be set to guarantee the controller running stably. Fortunately, we can take advantage of the cbench (Rotsos *et al.*, 2012) tool to measure the performance of the controller and obtain an appropriate utilization threshold of the CPU. PRE prefers a smaller period as long as the utilization of the CPU does not exceed this predefined threshold. Meanwhile, PRE would recheck the performance metric of a particular flow or link in a small time-scale later when its feature (such as the flow's transmission rate or link's residual bandwidth) changes massively. $\mathrm{UT_{cpu}}$ is used to represent the upper utilization threshold of the CPU. A lower CPU utilization threshold of the controller is defined by $\mathrm{LT_{cpu}}$ and the default polling period is T. To make the controller work with high performance, PRE maintains the CPU utilization of the controller in $[\mathrm{LT_{cpu}}, \mathrm{UT_{cpu}}]$, being adjacent to $\mathrm{UT_{cpu}}$ as far as possible. $\mathrm{CU_{cpu}}$ is adopted to describe the current CPU utilization of the controller, and the dynamic polling period adjustment is described in Algorithm 1.

PRE calculates the polling period for the next loop in $T/2$ later after receiving the first reply packet from OpenFlow switches in the current polling period. When the current CPU utilization satisfies $\mathrm{CU_{cpu}} < \mathrm{LT_{cpu}}$, PRE halves the polling period (line 3). PRE performs a precise adjustment when $\mathrm{LT_{cpu}} \leq \mathrm{CU_{cpu}} \leq \mathrm{UT_{cpu}}$. In this case, PRE gradually decreases the polling period by decreasing α to escape overload in the controller (line 5). Meanwhile, PRE adopts a parameter (dir) to opt for moving direction, which is related to the average RTT between the controller and the switches. Parameter dir is equal to 1 when RTT increases to 2× the average RTT; otherwise, dir is equal to -1. PRE sets the polling period to be $2T$ to alleviate the workload of the controller when $\mathrm{CU_{cpu}} > \mathrm{UT_{cpu}}$ (line 8).

For traffic prediction in DCNs, MicroTE (Benson *et al.*, 2011) shows that for nearly 70% of the ToR pairs the traffic remains stable for 1.5–2.5 s on average, and PRE predicts the transmitting rate (V) of a given flow in the next polling period based on this conclusion. Specifically, we assume that the previous query value is V_{p} and that the current query value is V_{c}. With regard to the fluctuation of the TCP sending rate, PRE regards that the flow stays in stable state if V_{c} is in the range $[0.9V_{\mathrm{p}}, 1.1V_{\mathrm{p}}]$ for per-flow transmitting rate prediction, and sets $(V_{\mathrm{p}}+V_{\mathrm{c}})/2$ as the prediction rate; otherwise, PRE believes that the flow suffers from congestion and would recheck the transmitting rate of the flow. If we assume that the new query value is V_{n}, then PRE sets $(V_{\mathrm{p}}+V_{\mathrm{c}}+V_{\mathrm{n}})/3$ as the prediction value. Therefore, per-flow's transmission rate is predicted according to

$$V = \begin{cases} (V_{\mathrm{p}}+V_{\mathrm{c}})/2, & 0.9V_{\mathrm{p}} \leq V_{\mathrm{c}} \leq 1.1V_{\mathrm{p}}, \\ (V_{\mathrm{p}}+V_{\mathrm{c}}+V_{\mathrm{n}})/3, & \text{otherwise.} \end{cases} \tag{4}$$

From Eq. (4), the transmission rate of each flow can be obtained, and then the values of $f(e)$ and W_{sd} can be easily calculated.

4.3 Performance-oriented resource allocation

PMPT is presented to solve this maximum multicommodity flow problem. PMPT must not only maximize resource utilization but also run in real time with low time complexity. Although the problem is NP-complete and has an exponential number of variables, it can be solved with a primal-dual algorithm.

Algorithm 1 Dynamic polling period adjustment

```
Input: $\mathrm{LT_{cpu}}$, $\mathrm{UT_{cpu}}$, and $T$
Output: $T$
 1: Obtain the current CPU utilization of the controller,
    $\mathrm{CU_{cpu}}$
 2: if $\mathrm{CU_{cpu}} < \mathrm{LT_{cpu}}$ then
 3:     $T \leftarrow 0.5T$
 4: else if $\mathrm{LT_{cpu}} \leq \mathrm{CU_{cpu}} \leq \mathrm{UT_{cpu}}$ then
 5:     $T \leftarrow T(1+\alpha \times \mathrm{dir})$ // dir = ±1 is the moving
        // direction and $\alpha$ is an adjustment factor
 6:     $\alpha \leftarrow \max\{0.01, \alpha - 0.01\}$
 7: else
 8:     $T \leftarrow 2T$
 9: end if
10: return $T$
```

We introduce dual variables $l(e)$ with each link capacity for constraint (1) and z_{sd} for demand constraint (2), where $l(e)$ is the cost of link and z_{sd} the lightest path from s to d. The dual can be written as follows:

$$\text{minimize} \quad \sum_{e \in E} b(e)l(e)$$

$$\text{s.t.} \begin{cases} \displaystyle\sum_{e \in p} l(e) \geq z_{sd}, \quad \forall e \in E, \ \forall p \in P_{sd}, & (5) \\ \displaystyle\sum_{s \in V} \sum_{d \in V} z_{sd} W_{sd} \geq 1, & (6) \\ l(e) \geq 0, \qquad \forall e \in E, & (7) \end{cases}$$

where $b(e)$ represents the available bandwidth of link e. Let L_{sd} denote the lightest path from s to d, which uses the link cost $l(e)$ of link e. The dual can then be rewritten as

$$\text{minimize} \quad \sum_{e \in E} b(e)l(e)$$

$$\text{s.t.} \begin{cases} \displaystyle\sum_{s \in V} \sum_{d \in V} L_{sd} W_{sd} \geq 1, & (8) \\ l(e) \geq 0, \quad \forall e \in E. & (9) \end{cases}$$

In other words, given any non-negative set of link costs $l(e)$, the upper bound of the dynamic routing problem is

$$\frac{\sum_{e \in E} b(e)l(e)}{\sum_{s \in V} \sum_{d \in V} W_{sd} L_{sd}}.$$

The fully polynomial time approximation scheme (FPTAS) can be used to solve this problem, which is simple to implement and runs significantly faster than a general linear programming solver in PMPT.

FPTAS provides the following performance guarantees: for any $\varepsilon > 0$, the solution has an objective function value within $(1+\varepsilon)$-factor of the optimal, and the running time is at most a polynomial function of the network size and $1/\varepsilon$. FPTAS in our case is a primal-dual algorithm, and some optimization is implemented on it to decrease the running time via dynamic graph algorithms (Madry, 2010). We optimize the primal-dual algorithm with the following processes to make it run faster and contribute less overhead on the controller:

1. In the FatTree network, given an arbitrary feasible traffic matrix, if a routing algorithm can evenly spread the traffic x_{ij} from server i to server j among all the possible uplinks at every layer, then all the links, including all the downlinks, are not overloaded, which has been verified in Cao *et al.* (2013). Meanwhile, for the FatTree network, once the spine switch is chosen, the routing path from the source to the destination is decided without ambiguity. Therefore, what just needs to be done is to precisely compute the routing from the source server to the spine switch, which can decrease the running time of the algorithm as well as the overhead of the controller. For VL2, a similar conclusion can be drawn according to the related algorithm presented in Cao *et al.* (2013).

2. Flows that are in the same ToR switch pair will compete for the same path as well as transmitting bandwidth in FatTree and VL2. So, LESSOR can put them into a group and compute all available paths for them only once, which will significantly decrease the overhead (complexity) of the controller.

3. In Madry (2010), the admissible-pair path should be computed periodically for each source-sink pair. To achieve this goal, a data structure is maintained that can provide an approximate shortest path for a flow group directly rather than compute everything from scratch, and this data structure is named the path dictionary. For any $s \to d$, the path dictionary is expressed as $P_{sd} \leftarrow \{B_a, \text{path1} : [b, \Delta b], \ldots\}$, where B_a represents the available bandwidth of the path with the heaviest load in P_{sd}, while b is the available bandwidth of a given path and Δb is the difference of available bandwidth between the path and the heaviest load path in P_{sd}. By taking advantage of this data structure, the multipath is computed just once for a given flow group. To be specific, the multipath is preferentially calculated and path information preserved in path dictionary for a given flow group; then we assign flows on each path until $\Delta b \leq 0$ rather than compute the lightest path for each flow, by which the overhead of the controller will be significantly decreased.

The primal-dual algorithm (Algorithm 2) for our problem works as follows. The algorithm first computes a value δ, which is the function of the desired accuracy level ε and the number of nodes n (lines 1–2), and then operates in phases where in each primal phase flow is routed along the approximately lightest reachable path. Specifically, for a given ToR pair, PMPT calculates their bisection bandwidth and sorts the flows according to their remaining FCT in ascending order based on the MRTF principle (line 5), and then puts the flow one by one with order on the path with lighter load based on the

Algorithm 2 Resource allocation

```
Input: $F_c = \{(s,d) : (f, B_f)\}$ // flow $f$ with its
    // demanded bandwidth $B_f$
Output: $\lambda, R$(Routing)
1: $\delta = (1+\varepsilon)/(n(1+\varepsilon))^{1/\theta}$, $D_L \leftarrow \delta$, and $l(e) \leftarrow \delta$ for
    $e \in E$ where $\theta = \varepsilon(1+\varepsilon)$
2: for each switch pair $s \to d$ do
3:     Calculate $W_{sd}$
4:     $B_{sd} \leftarrow 0$
5:     $F_{sd} \leftarrow F_c(s,d)$ // sort all the flows in $F_c(s,d)$
        // according to their remaining FCT in ascending
        // order
6:     $P_{sd} \leftarrow \{B_a, \text{path1} : [b, \Delta b], \ldots\}$
7:     $R \leftarrow \oslash$
8:     for each path $p$ in $P_{sd}$ do
9:         $d'(s) \leftarrow \Delta b = b - B_a$
10:        while $D_L > 0$ and $d'(s) > 0$ do
11:            for each flow $f$ in $F_{sd}$ do
12:                $b_f \leftarrow \min\{b, B_f\}$
13:                $R \leftarrow f$
14:                $d'(s) \leftarrow d'(s) - b_f$
15:                $B_{sd} \leftarrow B_{sd} + b_f$
16:                $b(e) \leftarrow b - b_f$
17:                $l(e) \leftarrow l(e)[1 + \varepsilon b(e)/c(e)]$
18:                Recalculate $D_L = \sum b(e)l(e)$
19:            end for
20:        end while
21:        $B_a \leftarrow B_a + d'(s)$
22:    end for
23: end for
24: return $\lambda = \max(B_{sd})/W_{sd}$, $R$
```

path dictionary (lines 8–13). For each flow f, its allocated bandwidth is the smaller one in $[b, B_f]$, where B_f is the demanded bandwidth of flow f. After calculating bandwidth for flow f, PMPT will update the related parameters, such as the cost $l(e)$ and remaining bandwidth $b(e)$ of link e (lines 14–18). This process of augmenting flow and updating the dual length is repeated until the problem is dual feasible.

With Algorithm 2, PMPT computes near-optimal routing and bandwidth reservation for each flow. Meanwhile, PMPT can obtain the value of λ, which is equal to $\max(B_{sd})/W_{sd}$. This algorithm follows the same vein as in Madry (2010). The running time of the primal-dual algorithm is $O(\varepsilon^{-2}mn\log m)$, where m is the number of links, and the final flow is a $(1-3\varepsilon)$-approximation to the optimal flow. In each iteration, Algorithm 2 computes the reachable path and the corresponding bandwidth provision for each flow to maximize resource utilization of the whole network. Unlike the algorithm in Madry (2010), the computation is organized on a ToR switch pair, and a precise path-computing algorithm is used for these grouped flows instead of a random path set. More importantly, PMPT allocates the bandwidth for many flows at the same time based on the path dictionary, which is totally different from the finding admissible pair algorithm in Madry (2010).

5 Evaluation

We evaluated the performance of LESSOR in the Mininet emulator (Handigol *et al.*, 2012), a high-fidelity network emulation framework built on Linux container based virtualization, which creates a virtual network running a real Linux kernel while the switches and applications code on a single machine. Its scale is smaller than that of production data center networks due to the single-machine CPU limitation (tens of Mb/s link bandwidth compared to 1 Gb/s). Mininet has been shown to faithfully reproduce implementation results from Al-Fares *et al.* (2010) and Alizadeh *et al.* (2010) with high fidelity (Handigol *et al.*, 2012), and has been used as a flexible testbed for networking experiments (Greenberg *et al.*, 2011). Both FatTree and VL2 were selected as the datacenter topologies in our simulation. We also implemented LESSOR in the testbed for further performance evaluation. We chose POX as our controller abiding by OpenFlow 1.0.0 specification and the performance of LESSOR was compared with that of TCP, ECMP, and Hedera under the web search workload. Each set of experiments was tested for 10 runs lasting 300 s, and the average value was designated as the experimental consequences for improving the accuracy of the experiments.

5.1 Methodology

5.1.1 Topology

The parameters of our test DCN topologies are set up according to Handigol *et al.* (2012). The experimental scenarios are devised as follows:

1. FatTree ($k=4$)

Parameter k in a FatTree represents the number of pods. There are 16 servers and 20 switches, and each ToR switch connects with two servers. The maximum number of parallel transmission paths for each end-to-end connection in different pods is four. The switch port buffer is set to be 150 packets, and the link bandwidth is 10 Mb/s.

2. VL2 ($d_0=2,\ d_1=4,\ d_2=4$)

Here d_0, d_1, and d_2 represent the numbers of 10 Gb/s Ethernet ports in each ToR switch, aggregation switch, and spine switch, respectively. There are 80 servers and 10 switches, and each ToR switch connects with 20 servers. The maximum number of parallel transmission paths for each end-to-end connection in different pods is four. The switch port buffer is arranged to be 150 packets, and the link bandwidth between servers and switches is 10 Mb/s, while the link bandwidth between switches is 100 Mb/s.

3. Testbed

The topology of our testbed is as shown in Fig. 3. There are eight servers and four switches, where the switches are Pica8 and servers are Lenovo desktop computers with 1 Gb/s NIC (network interface card). Each ToR switch connects with four servers, in which the maximum number of parallel transmission paths for each server pair is two. The bandwidth of each link is 1 Gb/s.

5.1.2 Benchmark workloads

Benson *et al.* (2010) and Chen *et al.* (2011) found that the traffic at the datacenter edge can be characterized by ON-OFF patterns. The ON and

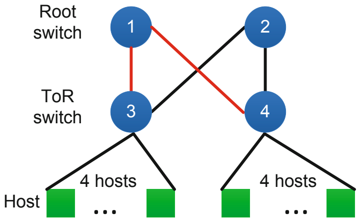

Fig. 3 Topology of the testbed

OFF periods, and packet interval time are drawn from three different log normal processes. The traffic follows a heavy-tailed distribution and is bursty. We consider one flow size distribution from a cluster running web search (Alizadeh *et al.*, 2010), and the workloads exhibit heavy-tailed characteristics with a mix of small and long flows. In the web search workload, over 95% of the bytes are from 30% of flows larger than 1 MB. In our experiments, the iperf traffic generation tool is adopted to generate ON-OFF flows on each server, which abides by the traffic pattern above.

5.1.3 Communication model

In datacenters, the communication patterns between servers include one-to-one, one-to-multi, and all-to-all communication models (Alizadeh *et al.*, 2010; Handigol *et al.*, 2012; Chiesa *et al.*, 2014). We use the random communication model, where each server chooses both the destination server and the number of destination servers in a random way, which includes all communication models above.

5.1.4 Schemes for comparison

The compared schemes include TCP, ECMP, and Hedera.

1. TCP

Standard TCP-New Reno is employed as the baseline of our evaluation. The initial window is set to 12 KB, and switches adopt Droptail. These are standard settings adopted in many studies (Alizadeh *et al.*, 2010).

2. ECMP

The standard ECMP algorithm (Ford *et al.*, 2013) is used, in which the controller chooses the next hop for each packet by hashing the five-tuple, including source/destination IP address, source/destination port number, and the protocol of transport layer. TCP-New Reno is employed in servers with the same parameters as in TCP.

3. Hedera

The code of Hedera is from Handigol *et al.* (2012). The polling time is set to be 5 s as Al-Fares *et al.* (2010) used, but the global first fit (GFF) is measured because simulated annealing is significantly more complex but does not provide much performance gain. Other parameters are the same as in TCP.

4. LESSOR

Our design has been described in Section 4 with settings $\mathrm{LT_{cpu}}$=50%, $\mathrm{UT_{cpu}}$=70%, T=5 s, and α=0.05. TCP-New Reno is also employed in servers, and the parameters are the same as in TCP.

5.2 Experimental results on Mininet

The performance of LESSOR is first tested under different DCN topologies and compared with that of TCP, ECMP, and Hedera under various loads. Figs. 4 and 5 illustrate the throughput and mean flow completion time of four different schemes in FatTree ($k = 4$) and VL2 under various loads, respectively.

5.2.1 Throughput

Figs. 4a and 4b show the throughput of TCP, ECMP, Hedera, and LESSOR in FatTree ($k = 4$) and VL2, respectively. It can be seen that LESSOR achieves the highest throughput in both FatTree and VL2 under various loads. With the increase in load, multipath schemes achieve apparent performance advantages over TCP. Hedera inquires long flow with high transmission bandwidth (more than 10% of link capacity, also named by elephant flow) from ToR switches and reroutes them by GFF to improve the load balancing. So, Hedera achieves higher throughput than ECMP. Compared with Hedera, LESSOR adopts active load balancing by installing flow tables for long flows based on PMPT, which can avoid more load unbalancing as well as network congestion and thus achieves higher throughput. Specifically, LESSOR provides 2.83%–80.1% and 2.3%–41.4% throughput improvement in FatTree and VL2, respectively. Meanwhile, we find that each scheme achieves better throughput in VL2 than in FatTree. This is because VL2 uses a 100 Mb/s link among switches, which can cope with more bursty traffic.

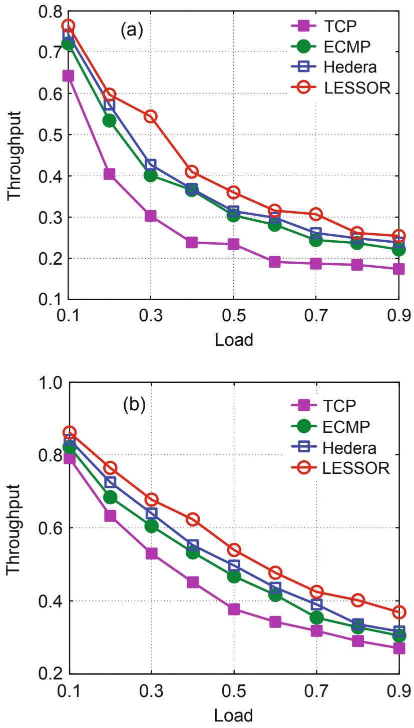

Fig. 4 Throughput in FatTree (k=4) (a) and VL2 (b) under different schemes

This conclusion has also been shown in Chiesa *et al.* (2014). Therefore, it can be concluded that LESSOR can efficiently improve network resource utilization.

5.2.2 Flow completion time

Figs. 5a and 5b show the mean flow completion time of TCP, ECMP, Hedera, and LESSOR in FatTree and VL2, respectively. It is seen that LESSOR obtains smaller FCT than other schemes, which demonstrates the advantage of LESSOR in resource excavation and allocation. Due to the single path available, flows in TCP suffer 18.2% FCT more than LESSOR; specifically, it works worse when the load is large. ECMP performs better than TCP, but it still cannot make full use of all the available bandwidth, so the mean FCT increases by 7.1%–26.4% compared with LESSOR. Hedera improves load balancing with dynamical scheduling, which decreases the FCT by 16.8% at most in FatTree and VL2 compared with ECMP. LESSOR performs best all the times and achieves higher performance improvement with the increase of load. It achieves 5.3%–32.5% decrease in FCT compared with other schemes. In conclusion, LESSOR can efficiently improve the throughput and decrease the FCT in both FatTree and VL2 under all loads.

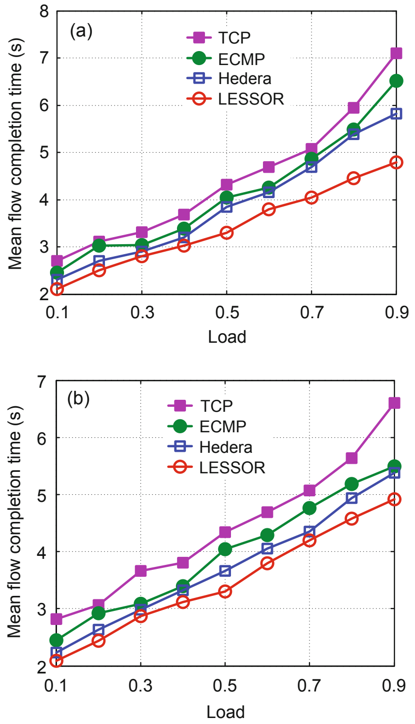

Fig. 5 Mean flow completion times in FatTree (k=4) (a) and VL2 (b) under different schemes

5.3 Experimental results on the testbed

In this subsection, the excellent performance of LESSOR is further verified by implementing it in a real testbed. We further discuss the scalability and complexity by comparing our method with Hedera (Al-Fares *et al.*, 2010), which is similar to our method in improving network throughput. Specifically, we use five metrics, i.e., throughput, FCT, 95th FCT, CPU utilization, and polling period, to comprehensively present the performance of our scheme. In Hedera, the polling time is set to be 5 s as in Al-Fares *et al.* (2010), but GFF alone is measured because the simulated annealing algorithm is significantly more complex but does not provide much performance gain.

5.3.1 Throughput

Fig. 6 presents the average throughput of TCP, ECMP, Hedera, and LESSOR. Obviously, LESSOR performs best under each load and improves the throughput by 2%–57.3% compared with the other

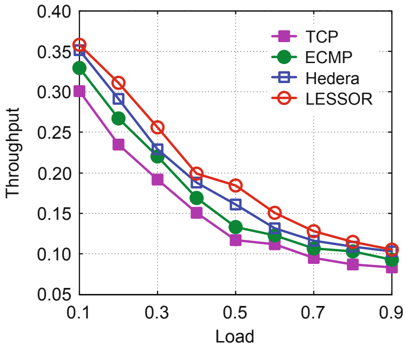

Fig. 6 Throughput of different schemes in the testbed

three schemes. Specifically, because there is only one path available in TCP, the throughput is worse than that of other three multipath schemes under each load. ECMP's hashing collision problem makes poor throughput. Hedera partially solves the hashing collision by recomputing another path for high throughput flows, but the GFF algorithm still produces load imbalance, and thus the throughput is lower than LESSOR. LESSOR can realize higher throughput owing to the PRE and PMPT algorithms, and prefers to put more flows on the lightest load path to achieve good load balancing.

5.3.2 Flow completion time

Fig. 7 illustrates the mean FCT and 95th FCT of TCP, ECMP, Hedera, and LESSOR under various loads. Generally speaking, LESSOR performs better in decreasing FCT compared with the other three schemes. As seen in the above analyses, because there is only one path available and the hashing collision problem, TCP and ECMP cannot make full use of all available bandwidth, and thus they perform worse than Hedera and LESSOR. Hedera achieves larger FCT and 95th FCT than LESSOR; specifically, it performs worse in terms of 95th FCT. Compared with LESSOR, Hedera adopts a reactive way in coping with load imbalance. Specifically, Hedera adopts ECMP to compute the routing for each flow at first glance, and reroutes elephant flows with GFF that it detects from ToR switches. On the contrary, LESSOR calculates the near-optimal path for each long flow to escape network congestion when it comes into the network, and thus achieves better load balancing, which contributes to higher throughput as well as smaller FCT compared with Hedera.

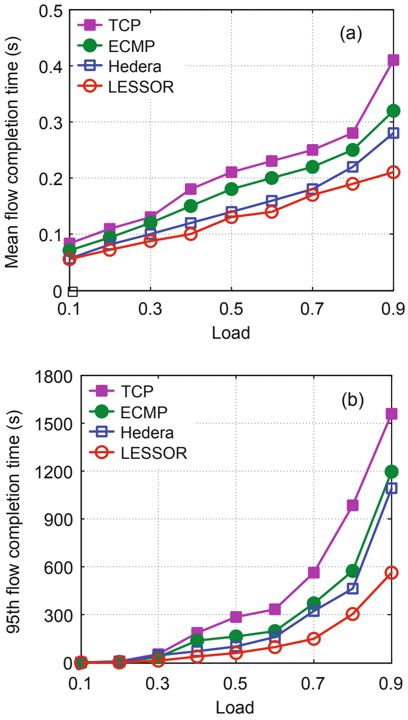

Fig. 7 Mean flow completion time (FCT) (a) and 95th FCT (b) of different schemes in the testbed

Meanwhile, the excellent performance of LESSOR relies on PRE, which makes full use of all available resources of the controller to obtain more fine-grained flow and link information for PMPT to execute better bandwidth allocation.

Fig. 8 illustrates the CPU utilization and polling period of Hedera and LESSOR under various loads. Hedera adopts static polling period (5 s in Al-Fares *et al.* (2010)), so it cannot make full use of the available resources of the controller to conduct more fine-grained traffic control. It can be seen from Fig. 8a that the CPU utilization is low when the load is less than 0.5. What is worse, the static polling period would impair the controller's performance in dealing with newly coming flows, because too much control message will cause large queuing delay when the load is heavy. In the testbed experiment, we find that more than 12% of flows fail to transmit at load=0.9 due to the failure of installing flow in time. LESSOR uses PRE, a dynamic polling period mechanism, which adapts its polling period based on the CPU utilization of the controller. From Fig. 8, we can learn that LESSOR uses a small polling period to obtain more fine-grained network information when the load is light. For example, the polling period of

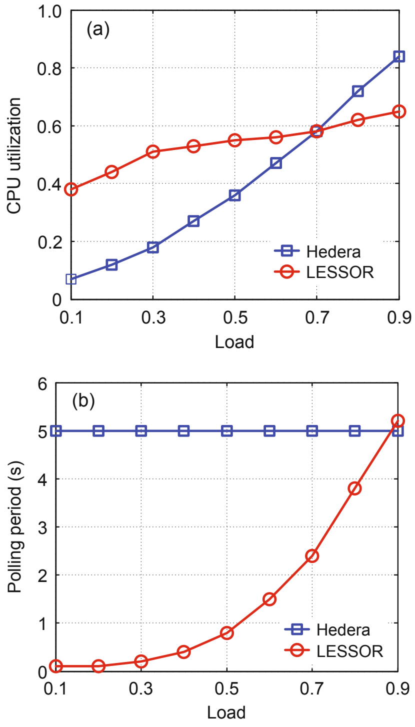

Fig. 8 CPU utilization (a) and polling period (b) of different schemes in the testbed

LESSOR is less than 1 s when the load is less than 0.5. Meanwhile, LESSOR can escape overload of the controller by enlarging the polling period. However, the polling period will not become very large unless the CPU utilization (or overhead) of the controller is higher than the upper threshold value all the time, but we can use a more powerful PC to act as the controller if this situation occurs. In our testbed experiment, the polling period is no more than 7.6 s even if the load is equal to 0.9. Moreover, LESSOR adopts a smaller polling period than Hedera even though their CPU utilizations are equal, which means that LESSOR uses the resource of the controller more efficiently. In conclusion, LESSOR not only improves network throughput but also maintains the stability of the whole network.

6 Related work

In this section, a brief description and contrast of our work with the most relevant research work will be presented.

6.1 Multipath transmission schemes

The existing multipath transmission schemes are divided into three categories: flow-level schemes such as ECMP, SPAIN (Mudigonda *et al.*, 2010), RRMP (Bredel *et al.*, 2014), and FlowBender (Kabbani *et al.*, 2014); packet-level schemes such as DRB (Cao *et al.*, 2013), RPS (Dixit *et al.*, 2013), and DeTail (Zats *et al.*, 2012); flowlet-level schemes such as MPTCP and CONGA (Alizadeh *et al.*, 2014). ECMP adopts a random load balancing mechanism, in which each router chooses the next hop for each packet by hashing the five-tuple, including source/destination IP address, source/destination port number, and the protocol of transport layer. SPAIN splits the multiple paths into different virtual local area networks (VLANs), and an offline network controller system precomputes and preinstalls the transmission path for each flow. RRMP simply assigns a new flow to the next available path. FlowBender is based on ECMP and uses end-host-driven rehashing to trigger dynamic flow-to-path assignment when detecting congestion. However, none of them can fully use the bisection bandwidth because random load balancing causes much collision. Resorting to the multi-address servers, MPTCP separates one (long) flow into multiple subflows and then delivers each subflow in different paths, and all subflows use a coupling congestion control algorithm (Raiciu *et al.*, 2011a). CONGA splits TCP flows into flowlets, estimates real-time congestion on fabric paths, and allocates flowlets to paths with small loads based on the feedback from remote switches. In DRB, for each outgoing packet, the source server selects one of the highest level switches (spine switches) based on the measured latency introduced by the end-server network stack to be the bouncing switch, and sends the packet to that switch. The bouncing switch then bounces the packet to the destination. Both MPTCP and DRB cause the disordering problem. RPS and DeTail place the burden of load balancing upon the switches themselves at the packet level, which requires hardware changes at the switches and thus makes it hard to deploy. LESSOR uses PMPT, a flow-level-based multipath transmission mechanism, to maximize bandwidth utilization by arranging each flow in an appropriate path according to global network view in a control period, which can get rid of packet disordering.

6.2 OpenFlow based schemes

There are many OpenFlow-based centralized schemes (Al-Fares *et al.*, 2010; Benson *et al.*, 2011; Curtis *et al.*, 2011a; 2011b) for solving the traffic engineering (TE) problems of DCNs. Unlike LESSOR, Hedera (Al-Fares *et al.*, 2010) adopts a static polling period and thus cannot obtain accurate information of the network state. MicroTE (Benson *et al.*, 2011) leverages a short-term and partial predictability of the traffic matrix to allocate new flow on the lightest path, while Mahout (Curtis *et al.*, 2011b) performs elephant flow classifications at the end-hosts by looking at the TCP buffer of outgoing flows, avoiding the need to invoke the controller for mice. However, they route flows with ECMP and try to decrease hash collision by detecting elephant flows, and this reactive method of improving load balancing will impair the network. DevoFlow (Curtis *et al.*, 2011a) extends data plane mechanisms of the switches with the objective of reducing the load toward the controller. LESSOR is more friendly for deployment because it does not require any modification to the switches and applications. Meanwhile, LESSOR tries to obtain more accurate flow statistics by making full use of the power of the controller and employs an effective resource allocation scheme, thus obtaining good performance.

7 Conclusions

This paper focused mainly on solving the issues of low resource utilization existing in DCNs. We proposed LESSOR, which adopts the OpenFlow-based centralized control mechanism and makes full use of the parallel paths in dense interconnection DCNs, to allocate the traffic to all the available paths near-optimally based on the network view maintained by the performance-oriented resource excavation mechanism.

Both theoretical analyses and experimental results showed that LESSOR can efficiently improve the resource utilization in DCNs. Moreover, the experimental results showed that LESSOR can increase the throughput significantly, exceeding that of TCP and ECMP by 9%–80.1% and 4.9%–38.3%, respectively. Besides, LESSOR outperformed other OpenFlow-based multipath forwarding solutions in achieving better load balancing and reducing AFCT. The experimental results also showed that LESSOR is scalable, and both the network scale and network load models have little impact on its performance. Therefore, LESSOR effectively addresses the datacenter TE issues. In future work, we will try to decrease the flow completion time with OpenFlow technology in DCNs.

References

Al-Fares, M., Loukissas, A., Vahdat, A., 2008. A scalable, commodity data center network architecture. *ACM SIGCOMM Comput. Commun. Rev.*, **38**(4):63-74. http://dx.doi.org/10.1145/1402946.1402967

Al-Fares, M., Radhakrishnan, S., Raghavan, B., *et al.*, 2010. Hedera: dynamic flow scheduling for data center networks. Proc. 7th USENIX Conf. on Networked Systems Design and Implementation, p.1-15.

Alizadeh, M., Greenberg, A., Maltz, D.A., *et al.*, 2010. Data center TCP (DCTCP). *ACM SIGCOMM Comput. Commun. Rev.*, **41**(4):63-74. http://dx.doi.org/10.1145/1851182.1851192

Alizadeh, M., Edsall, T., Dharmapurikar, S., *et al.*, 2014. CONGA: distributed congestion-aware load balancing for datacenters. *ACM SIGCOMM Comput. Commun. Rev.*, **44**(4):503-514. http://dx.doi.org/10.1145/2619239.2626316

Benson, T., Akella, A., Maltz, D.A., 2010. Network traffic characteristics of data centers in the wild. Proc. 10th ACM SIGCOMM Conf. on Internet Measurement, p.267-280. http://dx.doi.org/10.1145/1879141.1879175

Benson, T., Anand, A., Akella, A., *et al.*, 2011. MicroTE: fine grained traffic engineering for data centers. Proc. 7th Conf. on Emerging Networking EXperiments and Technologies, Article 8. http://dx.doi.org/10.1145/2079296.2079304

Bredel, M., Bozakov, Z., Barczyk, A., *et al.*, 2014. Flow-based load balancing in multipathed layer-2 networks using OpenFlow and multipath-TCP. Proc. 3rd Workshop on Hot Topics in Software Defined Networking, p.213-214. http://dx.doi.org/10.1145/2620728.2620770

Cao, J., Xia, R., Yang, P., *et al.*, 2013. Per-packet load-balanced, low-latency routing for Clos-based data center networks. Proc. 9th ACM Conf. on Emerging Networking EXperiments and Technologies, p.49-60. http://dx.doi.org/10.1145/2535372.2535375

Chen, Y., Jain, S., Adhikari, V.K., *et al.*, 2011. A first look at inter-data center traffic characteristics via Yahoo! datasets. Proc. IEEE INFOCOM, p.1620-1628. http://dx.doi.org/10.1109/INFCOM.2011.5934955

Chiesa, M., Kindler, G., Schapira, M., 2014. Traffic engineering with equal-cost-multipath: an algorithmic perspective. Proc. IEEE Conf. on Computer Communications, p.1590-1598. http://dx.doi.org/10.1109/INFOCOM.2014.6848095

Curtis, A.R., Mogul, J.C., Tourrilhes, J., *et al.*, 2011a. DevoFlow: scaling flow management for high-performance networks. *ACM SIGCOMM Comput. Commun. Rev.*, **41**(4):254-265. http://dx.doi.org/10.1145/2043164.2018466

Curtis, A.R., Kim, W., Yalagandula, P., 2011b. Mahout: low-overhead datacenter traffic management using end-host-based elephant detection. Proc. IEEE INFOCOM, p.1629-1637. http://dx.doi.org/10.1109/INFCOM.2011.5934956

Dixit, A., Prakash, P., Hu, Y.C., *et al.*, 2013. On the impact of packet spraying in data center networks. Proc. IEEE INFOCOM, p.2130-2138. http://dx.doi.org/10.1109/INFCOM.2013.6567015

Ford, A., Raiciu, C., Handley, M., *et al.*, 2013. TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824. http://dx.doi.org/10.17487/RFC6824

Greenberg, A., Hamilton, J.R., Jain, N., *et al.*, 2011. VL2: a scalable and flexible data center network. *Commun. ACM*, **54**(3):95-104. http://dx.doi.org/10.1145/1897852.1897877

Handigol, N., Heller, B., Jeyakumar, V., *et al.*, 2012. Reproducible network experiments using container-based emulation. Proc. 8th Int. Conf. on Emerging Networking EXperiments and Technologies, p.253-264. http://dx.doi.org/10.1145/2413176.2413206

Hong, C.Y., Caesar, M., Godfrey, P.B., 2012. Finishing flows quickly with preemptive scheduling. *ACM SIGCOMM Comput. Commun. Rev.*, **42**(4):127-138. http://dx.doi.org/10.1145/2377677.2377710

Hopps, C.E., 2000. Analysis of an Equal-Cost Multi-path Algorithm. RFC 2992. Available from http://www.ietf.org/rfc/rfc2992.txt.

Jain, S., Kumar, A., Mandal, S., *et al.*, 2013. B4: experience with a globally-deployed software defined WAN. *ACM SIGCOMM Comput. Commun. Rev.*, **43**(4):3-14. http://dx.doi.org/10.1145/2534169.2486019

Kabbani, A., Vamanan, B., Hasan, J., *et al.*, 2014. FlowBender: flow-level adaptive routing for improved latency and throughput in datacenter networks. Proc. 10th ACM Int. Conf. on Emerging Networking EXperiments and Technologies, p.149-160. http://dx.doi.org/10.1145/2674005.2674985

Le, Q.Q., Yang, G.W., Hung, W.N.N., *et al.*, 2014. Performance-driven assignment and mapping for reliable networks-on-chips. *J. Zhejiang Univ.-Sci. C (Comput. & Electron.)*, **15**(11):1009-1020. http://dx.doi.org/10.1631/jzus.C1400055

Li, X.L., Wang, H.M., Guo, C.G., *et al.*, 2012. Topology awareness algorithm for virtual network mapping. *J. Zhejiang Univ.-Sci. C (Comput. & Electron.)*, **13**(3): 178-186. http://dx.doi.org/10.1631/jzus.C1100282

Madry, A., 2010. Faster approximation schemes for fractional multicommodity flow problems via dynamic graph algorithms. Proc. 42nd ACM Symp. on Theory of Computing, p.121-130. http://dx.doi.org/10.1145/1806689.1806708

McKeown, N., Anderson, T., Balakrishnan, H., *et al.*, 2008. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.*, **38**(2):69-74. http://dx.doi.org/10.1145/1355734.1355746

Mudigonda, J., Yalagandula, P., Al-Fares, M., *et al.*, 2010. SPAIN: COTS data-center Ethernet for multipathing over arbitrary topologies. Proc. 7th USENIX Conf. on Networked Systems Design and Implementation, p.18-33.

Peng, Y., Chen, K., Wang, G., *et al.*, 2014. HadoopWatch: a first step towards comprehensive traffic forecasting in cloud computing. Proc. IEEE Conf. on Computer Communications, p.19-27. http://dx.doi.org/10.1109/INFOCOM.2014.6847920

Qi, H., Shiraz, M., Liu, J.Y., *et al.*, 2014. Data center network architecture in cloud computing: review, taxonomy, and open research issues. *J. Zhejiang Univ.-Sci. C (Comput. & Electron.)*, **15**(9):776-793. http://dx.doi.org/10.1631/jzus.C1400013

Raiciu, C., Handley, M., Wischik, D., 2011a. Coupled Congestion Control for Multipath Transport Protocols. RFC 6356. http://dx.doi.org/10.17487/RFC6356

Raiciu, C., Barre, S., Pluntke, C., *et al.*, 2011b. Improving datacenter performance and robustness with multipath TCP. *ACM SIGCOMM Comput. Commun. Rev.*, **41**(4):266-277. http://dx.doi.org/10.1145/2043164.2018467

Rotsos, C., Sarrar, N., Uhlig, S., *et al.*, 2012. OFLOPS: an open framework for OpenFlow switch evaluation. Proc. 13th Int. Conf. on Passive and Active Measurement, p.85-95. http://dx.doi.org/10.1007/978-3-642-28537-0_9

Wilson, C., Ballani, H., Karagiannis, T., *et al.*, 2011. Better never than late: meeting deadlines in datacenter networks. *ACM SIGCOMM Comput. Commun. Rev.*, **41**(4):50-61. http://dx.doi.org/10.1145/2018436.2018443

Yu, C., Lumezanu, C., Zhang, Y., *et al.*, 2013. FlowSense: monitoring network utilization with zero measurement cost. Proc. 14th Int. Conf. on Passive and Active Measurement, p.31-41. http://dx.doi.org/10.1007/978-3-642-36516-4_4

Zats, D., Das, T., Mohan, P., *et al.*, 2012. DeTail: reducing the flow completion time tail in datacenter networks. *ACM SIGCOMM Comput. Commun. Rev.*, **42**(4):139-150. http://dx.doi.org/10.1145/2377677.2377711