Received: 1 April 2018 | Revised: 25 September 2018 | Accepted: 2 October 2018

DOI: 10.1002/dac.3843

RESEARCH ARTICLE

WILEY

SDN-based dynamic multipath forwarding for inter–data center networking

Yao-Chun Wang[1] | Ying-Dar Lin[1] | Guey-Yun Chang[2]

[1] College of Computer Science, National Chiao Tung University, Hsinchu, Taiwan

[2] Department of Computer Science, National Central University, Taoyuan, Taiwan

Correspondence
Yao-Chun Wang, College of Computer Science, National Chiao Tung University, Hsinchu 30010, Taiwan.
Email: scott0612@cht.com.tw

Summary

Equal-cost multipath (ECMP)–based traffic engineering (TE) methods are commonly used in intra–data center (DC) networks to improve the transmission performance for east-west traffic (ie, traffic from server to server within a DC). However, applying ECMP on inter-DC wide area network (WAN) offers limited performance enhancement as a result of irregular network topology. Since TE can be intelligently and efficiently realized with software-defined networking (SDN), SDN-based multipath becomes a popular option. However, SDN suffers from scalability issue caused by limited ternary content-addressable memory (TCAM) size. In this paper, we propose an SDN-based TE method called dynamic flow-entry-saving multipath (DFSM) for inter-DC traffic forwarding. DFSM adopts source-destination–based multipath forwarding and latency-aware traffic splitting to reduce the consumption of flow entries and achieve load balancing. The evaluation results indicate that DFSM saves 15% to 30% of system flow entries in practical topologies and reduces the standard deviation of path latencies from 10% to 7% than do label-switched tunneling, and also reduces average latency by 10% to 48% by consuming 6% to 20% more flow entries than do ECMP in less-interconnected topologies. Note that the performance gain may not always be proportional to flow entry investment, with the interconnectivity between nodes being an important factor. The evaluation also indicates that per-flow provision consumes several times the flow entries consumed by DFSM but reduces latency by 10% at most. Besides, DFSM reduces the standard deviation of path latencies from 14% to 7% than do even traffic splitting.

KEYWORDS

data center interconnect, multipath, OpenFlow, SDN

1 | INTRODUCTION

Internet service provider (ISP) nowadays builds clouds to deliver various online services. In recent trends, a cloud often consists of multiple data centers (DCs) located in different geographic areas, supporting application data exchanges and remote backup, as well as regional service. For intra-DC networks, Fat-tree[2] (or leaf-spine) is commonly adopted for

A shorter version of this paper appeared in the Proceedings of the IEEE LANMAN 2017.[1]

fault tolerance and reducing implementation costs. And for inter-DC networks, a large-scale cloud often connects multiple DCs via a wide area network (WAN).

1.1 | Multipath routing: intra-DC vs inter-DC

Many services in the cloud rely on low-latency DC interconnect (DCI) to enhance user experience (eg, streaming and web service); because of large and rapidly growing inter-DC traffic, traffic engineering (TE) methods are widely used to enhance inter-DC WAN performance. Multipath routing is a common way for TE; it, compared with single path routing, enhances bandwidth utilization and mitigates congestion. Traditional networks use Open Shortest Path First (OSPF). If there exists more than one shortest path between a node pair, the source node will split traffic into flows by a hash function and map the flows to these paths for transmission according to an equal-cost multipath (ECMP) mechanism.[3] For an intra-DC network with leaf-spine topology, the number of shortest paths between each leaf node pair is equal to the number of spine nodes in the network, and therefore, ECMP-based TE methods[4-7] are commonly used in intra-DC networks to improve the transmission performance for east-west traffic. However, the number of shortest paths between each node pair varies with irregular topologies, and applying ECMP on inter-DC WAN offers limited performance enhancement when there are few or no extra paths to share the traffic load.

1.2 | Software-defined inter-DC WAN

Software-defined networking (SDN) decouples the control plane and the data (forwarding) plane, relying on a centralized controller to collect global network information and to forward flows in the network, by setting the forwarding rules to switches. Compared with traditional multi-protocol label switching IP/MPLS-based TE, SDN-based TE[8] can be much more intelligently and efficiently realized thanks to the centralized global view of complicated networks and flexible flow control ability. An SDN-based TE approach simplifies per-flow- or application-based quality of service (QoS) provisioning. For example, FlowQoS[9] performs on-the-fly application identification for each flow of users of a broadband access network, and it forwards individual flows according to user-specified priorities for those applications. While a per-flow approach brings finer granularity, it suffers from a scalability issue. OpenFlow[10] is a popular standard for SDN. In an OpenFlow deployment, switches rely on ternary content-addressable memory (TCAM)[11] to store flow entries (forwarding entries), which results in TCAM size becoming the bottleneck of SDN. To mitigate this scalability issue, a differentiated services (DiffServ) model[12] divides the incoming flows into several classes and treats packets in the same class identically (ie, to the same set of tunnel paths). Both Google[13] and Microsoft[14] deployed software-defined inter-DC WAN, mapping site-to-site traffic to a set of tunnels to maximize the network utilization.

Although SDN has the potential to substantially simplify network management, SDN may be deployed incrementally in an existing network along with legacy network devices, because of the budget constraints.[15,16] Partial SDN deployment, which is also called hybrid SDN, is increasingly considered as a transition phase towards full SDN deployment. In this transition phase, maintenance of legacy devices still depends on manual operation and experience. Several researches[17-19] focus on TE in SDN/OSPF hybrid network; these approaches mainly adjust the OSPF weights or inject fake nodes/links into OSPF protocol to alter the forwarding behavior of legacy devices, and they also split flows at a fraction of network nodes, ie, SDN nodes.

As more and more SDN-centric network operating systems (eg, PICA8,[20] Switch Light OS,[21] and Open Network Linux[22]) and white box switch vendors (eg, Juniper, Dell, Edge-Core, and Quanta) emerge, SDN deployment is no longer a high cost. In this paper, we consider TE in fully deployed SDN.

1.3 | Scalability of inter-DC WAN

Central Office Re-architected as a Datacenter (CORD)[23] integrates SDN, network function virtualization (NFV), and Cloud technologies to transform edge Telco Central Offices into agile DCs, enabling network operators to deliver innovative services. In future years, the number of DCs will explode because of this trend (for example, AT&T currently operates 4700 Central Offices), and so does the consumption of flow entries in software-defined inter-DC WAN. Consequently, the scalability issue should be carefully considered.

1.4 | The proposed method

In this paper, we propose an SDN-based TE method called dynamic flow-entry-saving multipath (DFSM) for traffic forwarding in software-defined inter-DC WAN to meet latency demands of DCI services between DC pairs with fewest flow entries. DFSM has the following features.

1.4.1 | Flow-entry-saving multipath

For flow entry saving, DFSM uses source-destination–based forwarding, ie, a forwarding node forwards packets to possible next hops, on the basis of both packet's source and destination addresses (eg, IP). With source-destination–based forwarding, traffic splitting can occur at every forwarding node. Compared with popular label-switched tunneling[14] (ie, a source node scatters the traffic over multiple tunnels, and each forwarding node forwards each packet along a tunnel on the basis of packet's label pushed by the source node), source-destination–based forwarding merges per-tunnel flow entries and splits traffic at intersection-forwarding nodes (or split points) crossed by multiple paths. For ease of discussion, every intersection node of two paths is also called a split point.

1.4.2 | Intra-DC-pair fairness

The most common implementation of flow-based path selection relies on hash functions; since the result of path selection is randomly determined by hash, the latency experienced by each flow is unpredictable. In order to achieve better flow latency control, DFSM offers equal-latency paths for each DC pair by latency-aware traffic splitting, so as to prevent flows of the same DC pair from experiencing widely different latencies.

1.4.3 | Inter-DC-pair fairness

DFSM allocates available paths to each unsatisfied DC pair that cannot reach its latency demand until all DC pairs are satisfied, or nothing can be improved. For inter-DC-pair fairness, DFSM minimizes the difference of satisfaction degrees of DC pairs.

In this paper, we evaluate DFSM in terms of the performance of all DC pairs and the consumption of system flow entries, on the basis of simulation traffic of DC pairs. The following issues are also investigated: (1) the comparison of the consumption of system flow entries and average path latency between the adoption of DFSM, label-switched tunneling, ECMP, and per-flow provision; and (2) the comparison of standard deviation of path latencies between the adoption of DFSM, source-destination–based forwarding with even traffic splitting, and label-switched tunneling with latency-aware traffic splitting.

The contribution of this paper is summarized below. First, we proposed a TE method called DFSM, which saves flow entries while executing dynamic multipath adjustment in software-defined inter-DC WAN for satisfying latency demand of DCI service between each DC pair. Second, we considered both inter-DC-pair fairness and intra-DC-pair fairness. Third, we evaluate DFSM in terms of performance and the consumption of system flow entries on several practical topologies, and the results indicate that DFSM provides better flow entry investment returns.

The remainder of this paper is organized as follows: Section 2 discusses related works, Section 3 describes the notations and problem statement, Section 4 illustrates solution details, Section 5 gives an analysis of the flow entry saving, Section 6 shows the experimental results and related observation, and Section 7 contains the conclusions.

2 | RELATED WORK

2.1 | Traffic splitting

Packet-based splitting[4] and flow-based splitting[24] are widely adopted traffic splitting strategies. Packet-based splitting is known to provide better load balancing via finer-grained traffic control; however, the latency varies widely from path to path in an inter-DC WAN because of irregular topology, and out-of-order delivery of transmission control protocol (TCP) packets may easily occur. In order to prevent such out-of-order delivery, flow-based splitting is adopted in DFSM.

2.2 | Multipath WAN forwarding

Table 1 gives a summary of related works. WL2[25] uses a single L2 tunnel path to connect a DC pair on top of the Internet, and it adopts a hierarchical addressing scheme to reduce flow entries used in both intra-DC and inter-DC networks. However, L2 tunneling still relies on underlay routing; WL2 does not focus on performance enhancement on inter-DC networks. Hybrid Routing[26] finds a near-optimal routing combination of destination-based routing and explicit routing (traffic splitting) to minimize the maximum link utilization in a network. Since Hybrid Routing calculates routing on the basis of multiple given traffic matrices and does not mention dynamic adjustment, it may be unsuitable for inter-DC WAN with unpredictable and fast-changing traffic. wECMP-d[27] replaces even traffic splitting of ECMP with delay-sensitive traffic splitting according to given traffic demands to minimize overall end-to-end delays, but the adoption of equal-cost shortest paths may result in an inability to fully utilize link resources in irregular topologies. B4[13] and SWAN[14] both adjust the traffic sending rate of each edge DC, and they split inter-DC traffic at source edge of a DC pair to scatter the traffic over multiple tunnel paths in order to maximize the utilization of network links.

B4[13] uses IP in IP tunnel over network running border gateway protocol (BGP) / intermediate system-to-intermediate system (ISIS) to achieve worldwide deployment. SWAN[14] uses label-switched tunneling in SDN; however, these tunnel paths may cross at some forwarding nodes, at which time per-tunnel flow entry is required at these intersection-forwarding nodes to forward packets of the DC pair. The number of flow entries needed on each forwarding node increases as the number of DCs connected by inter-DC WAN increases; how to reduce flow entries needed by each DC pair while satisfying performance demand of each DC pair is the key to providing scalability.

In summary, for multipath selection, the adoption of *k* shortest/minimum cost paths provides better link utilization than do ECMP in irregular topologies. For path provision and traffic splitting, source-destination–based forwarding, compared with destination-based routing, outperforms label-based switching in flow entry saving and has the ability to provide DiffServ for different sources.

3 | PROBLEM STATEMENT

Enterprise customers often have their own service level agreement (SLA) for their DCI services. In the light of this, we consider heterogeneous latency demands for inter-DC services and suppose that every DC pair has its own bounded latency demand range.

The main objective of DFSM is to satisfy the latency demand of DCI service between each DC pair while minimizing the consumption of flow entries. We also consider inter-DC-pair fairness and intra-DC-pair fairness. For inter-DC-pair fairness, we minimize the difference of satisfaction degrees of DC pairs. Here, the satisfaction degree of a DC pair is

TABLE 1 Related works

Work	Objective	Path Provision	Multipath Selection	Traffic Splitting	Path Adjustment
WL2[25]	Achieve scalable L2	Overlay (L2)	No	No	No
B4[13]	Maximize average bandwidth	Overlay (L3)	Minimum cost paths without bottleneck edges	At only source node	Yes
SWAN[14]	Maximize network utilization	Switching (label-switched tunneling)	*k* shortest paths	At only source node	Yes
Hybrid Routing[26]	Minimize maximum link utilization	Routing (destination-based and explicit routing)	Finding the percentage of traffic demand on each link	At possible nodes	Not mentioned
wECMP-d[27]	Minimize overall end-to-end delay	Routing (destination-based)	ECMP	At possible nodes	Not mentioned
DFSM	Satisfy each latency demand	Routing (source-destination–based)	*k* shortest paths	At possible nodes	Yes

Abbreviations: DFSM, dynamic flow-entry-saving multipath; ECMP, equal-cost multipath.

defined to be the latency demand upper bound divided by the average path latency. For intra-DC-pair fairness, we minimize path latency difference among all paths allocated for each DC pair.

3.1 | Notation and definition

Table 2 gives the notations. We first model the link latency t_{ij}. With given bandwidth μ_{ij} and measured load λ_{ij} on $link_{ij}$, we use M/M/1 queuing model[28] to derive latency t_{ij} on $link_{ij}$, so t_{ij} is given by

$$t_{ij} = 1/\left(\mu_{ij} - \lambda_{ij}\right).$$

For a DC pair (s_s, s_d), there may exists many simple paths between s_s and s_d. Note that both s_s and s_d are access points of the network topology G and have a directed link with the border gateway of distinct DC. A simple path $path_{sd}$ usually includes many links, so that the latency t_path_{sd} of $path_{sd}$ is the sum of the latency of each link on $path_{sd}$, and thus t_path_{sd} is given by

$$t\_path_{sd} = \Sigma t_{ij}, \text{ where } link_{ij} \in path_{sd}.$$

Supposing that n_{sd} possible flow paths are used by a DC pair (s_s, s_d), we define the average path latency $avg\_t\_path_{sd}$ as

$$avg\_t\_path_{sd} = 1/n_{sd}{}^{*}\, \Sigma t\_path_{sd}.$$

The latency demand range of a DC pair (s_s, s_d) is bounded by $threshold_{sd}{}^{\mathrm{L}}$ and $threshold_{sd}{}^{\mathrm{U}}$, depending on the traffic type. For example, the demand values of $threshold_{sd}{}^{\mathrm{L}}$ and $threshold_{sd}{}^{\mathrm{U}}$ for the DC pair, which provides latency-sensitive services (eg, web service), should be both lower than the demand values for the DC pair, which provides latency-tolerant services (eg, data replication).

To satisfy the latency demand of each DC pair (s_s, s_d) (ie, make the $avg\_t\_path_{sd}$ within the range between $threshold_{sd}{}^{\mathrm{L}}$ and $threshold_{sd}{}^{\mathrm{U}}$), we adjust paths for each DC pair globally. To achieve intra-DC-pair fairness for a DC pair (s_s, s_d), we minimize the standard deviation σ_{sd} of path latencies among n_{sd} allocated paths, by adjusting the split ratio $\alpha^{sd}{}_{ij}$ at each intersection-forwarding node (or split point) crossed by more than one $path_{sd}$. Note that the sum of split ratios of each split point is equal to 1.

TABLE 2 Notations

Category	Notation	Description
Topology	$G = (S, L)$	The network topology with switches S and links L
Switch	s_i	The ith switch ($s_i \in S$)
Link	$link_{ij}$ μ_{ij}, λ_{ij}, t_{ij}	The directed link from s_i to s_j (s_i, $s_j \in S$; $link_{ij} \in L$; $i \neq j$) The bandwidth, offered load, and latency of $link_{ij}$
Path	$path_{sd}$ t_path_{sd}	A simple path from s_s to s_d (s_s, $s_d \in S$; $s \neq d \neq d$) The latency of $path_{sd}$
Decision variable	n_{sd} $\alpha_{ij}{}^{sd}$	The number of possible flow paths used for a pair (s_s, s_d) The split ratio of traffic from s_s to s_d on $link_{ij}$ for a split point s_i
Multipath	$avg\_t\_path_{sd}$ σ_{sd} $flow\ entry_{sd}$	The average latency of n_{sd} used paths The standard deviation of latencies of n_{sd} used paths The total number of flow entries used for a pair (s_s, s_d)
Demand	$threshold_{sd}{}^{\mathrm{L}}$ $threshold_{sd}{}^{\mathrm{U}}$	The lower bound of demand on average path latency for a pair (s_s, s_d) The upper bound of demand on average path latency for a pair (s_s, s_d)

3.2 | Problem description

With these notations, the objective of DFSM can be described as follows: Given an inter-DC WAN with topology $G = (S, L)$, the latency t_{ij} of each $link_{ij}$ belonging to L, the goal is to allocate paths for each DC pair, which minimizes the number of flow entries $flow\ entry_{sd}$ and the standard deviation σ_{sd} for each DC pair (s_s, s_d) subject to $threshold_{sd}^{L} < avg\_t\_path_{sd} < threshold_{sd}^{U}$, and also minimizes the difference of satisfaction degrees of DC pairs.

4 | DYNAMIC FLOW-ENTRY-SAVING MULTIPATH

DFSM is implemented as a centralized control plane application that manages multiple OpenFlow controllers (each responsible for some part of the network), controlling all OpenFlow switches in a fully deployed SDN environment. DFSM periodically monitors the load of each link by port statistics (tx_bytes/rx_bytes) sampling via OpenFlow controllers and computes link latency with given link bandwidth by M/M/1 queuing model. With derived link latencies, DFSM periodically adjusts paths allocated for DC pairs according to their demands. In addition to periodic path adjustments, a path adjustment and a new path finding will also be triggered when there is a newly added DC pair (ie, new performance demand).

To identify whether a DC pair reaches its latency demand, a demand fulfillment status is recorded for each DC pair. There are three possible demand fulfillment statuses: *oversatisfied* (ie, $avg\_t\_path_{sd} < threshold_{sd}^{L}$), *satisfied* (ie, $threshold_{sd}^{L} < avg\_t\_path_{sd} < threshold_{sd}^{U}$), and *unsatisfied* (ie, $threshold_{sd}^{U} < avg\_t\_path_{sd}$). To avoid transient congestion during network updates, the main idea of DFSM is to remove paths for oversatisfied DC pairs and to allocate available paths to unsatisfied DC pairs.

DFSM consists of the following components: flow-entry-saving multipath, intra-DC-pair fairness, and inter-DC-pair fairness.

4.1 | Flow-entry-saving multipath

4.1.1 | Source-destination-based forwarding

Compared with label-switched tunneling[14] (ie, a source node scatters the traffic over multiple tunnels, and each forwarding node forwards each packet along a tunnel on the basis of packet's label pushed by the source node), source-destination–based forwarding merges per-tunnel flow entries on intersection-forwarding nodes crossed by multiple paths. Figure 1 shows the differences between these two forwarding methods. For label-switched tunneling, split point s_i matches packets from each tunnel between the DC pair and keeps packets from being transferred along the same tunnel (with label swapping), which means the number of flow entries at s_i is equal to the number of paths passing through it. For source-destination–based forwarding, the split point s_i matches packets with both its source and destination address and sends each flow to next hop by hash. Therefore, the split point only needs one flow entry for each DC pair.

According to OpenFlow,[10] a flow entry can point to a group entry to perform additional forwarding behaviors (eg, load balancing and fast failover). Each group entry has action buckets. If "select" group type is used, packets will be processed by a single bucket in the action buckets, on the basis of a switch-computed selection algorithm (eg, hash) and bucket weights. Since group entries are responsible for only the execution of actions, group entries are commonly stored in static random-access memory (SRAM).

The traffic splitting of DFSM is implemented by the group entry with the "select" type. All possible routing paths can be maintained in a group entry at each split point. For a DC pair, each split point only needs one flow entry, which points to a group entry to split traffic flows to multiple next hops, thereby saving flow entries. According to the theoretical analysis in Section 5, the more intersection nodes between paths, the more flow entries DFSM can save.

Note that the cycles between the selected paths will cause "traffic looping" under the source-destination–based forwarding, resulting in a portion of the traffic never reaching its destination. In order to prevent "traffic looping," DFSM eliminates improper paths by cycle detection[29] while computing available paths for new-added DC pairs.

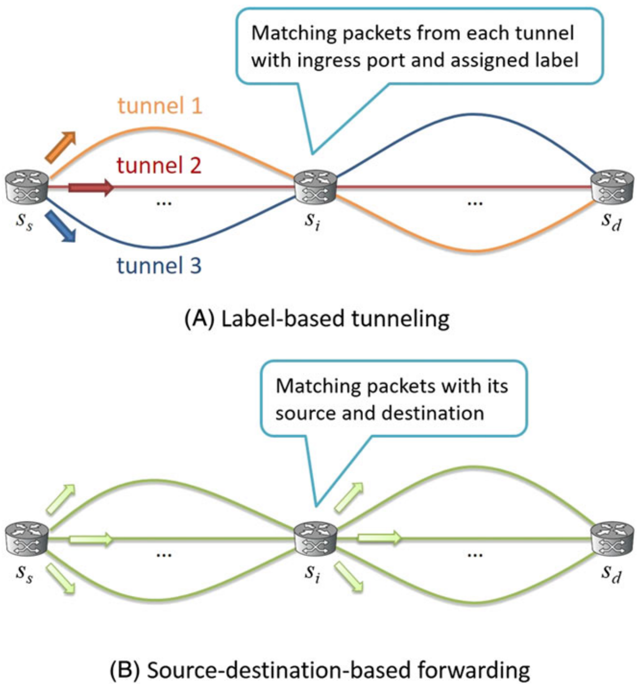
Matching packets from each tunnel with ingress port and assigned label
tunnel 1
tunnel 2
tunnel 3
s_s
s_i
s_d
(A) Label-based tunneling
Matching packets with its source and destination
s_s
s_i
s_d
(B) Source-destination-based forwarding

FIGURE 1 A, Label-based tunneling. B, Source-destination–based forwarding

4.1.2 | Greedy path selection

To meet latency demands with fewer paths (lower TCAM requirements), DFSM assigns the least-latency paths to unsatisfied DC pairs. To reduce path computation overhead, DFSM computes available k-least-hop-count paths[30] only for newly added DC pairs. During periodic path adjustments, DFSM finds the least-latency paths for path allocation by determining the latencies of these precomputed available paths, instead of recomputing paths every time.

4.2 | Intra-DC-pair fairness

To achieve intra-DC-pair fairness by offering equal-latency paths to each DC pair, DFSM takes path latencies into consideration and decides split ratios at each split point, so as to balance the traffic loads (or latencies) among multiple paths assigned to each DC pair.

The main idea of latency-aware traffic splitting in DFSM is to split traffic at intersection-forwarding nodes and to be applicable to non–disjoint-edge paths. As in previous literature,[7,14] the basic idea of multipath load balancing in DFSM is to have the split ratio become inversely proportional to the path load. However, the previous approaches split traffic for a node pair only at source node[7,14] and are designed only for edge-disjoint paths.[7]

In DFSM, for a split point, the split ratio corresponding to a specific next hop is inversely proportional to the average latency of the path segments that start at the split point and proceed via the specific next hop to the destination. Figure 2

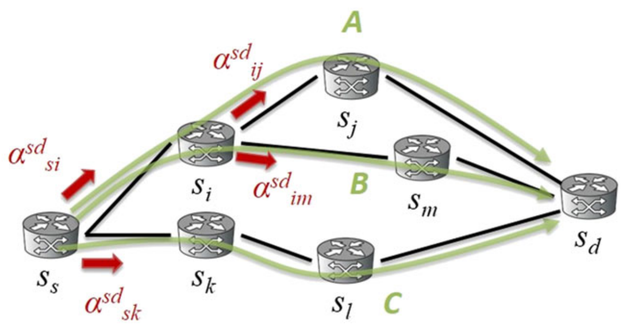

FIGURE 2 An example used to illustrate the split ratio

shows an example used to illustrate the split ratio computation in DFSM. There are three paths—A [s_s, s_i, s_j, s_d], B [s_s, s_i, s_m, s_d], and C [s_s, s_k, s_l, s_d]—used by a DC pair (s_s, s_d). In this case, s_s and s_i are both split points, with the next hops [s_i, s_k] and [s_j, s_m], respectively.

To simplify the illustration, the latency of the path segment [s_i, s_j, s_d] of A is denoted by $t\_path_{id}{}^{A}$, and the latency of the path segment [s_i, s_m, s_d] of B is denoted by $t\_path_{id}{}^{B}$. For split point s_i, the split ratio to s_j, ie, $\alpha_{ij}{}^{sd}$, is inversely proportional to $t\_path_{id}{}^{A}$; similarly for the split ratio to s_m, ie, $\alpha_{im}{}^{sd}$, is inversely proportional to $t\_path_{id}{}^{B}$. To be specific, we denote the least common multiple of $t\_path_{id}{}^{A}$ and $t\_path_{id}{}^{B}$ by t_lcm_{id}. Then $\alpha_{ij}{}^{sd}$ can be expressed as

$$\alpha_{ij}{}^{sd} = \left(t\_\mathrm{lcm}_{id}/t\_path_{id}{}^{A}\right)/\left(\left(t\_\mathrm{lcm}_{id}/t\_path_{id}{}^{A}\right) + \left(t\_\mathrm{lcm}_{id}/t\_path_{id}{}^{B}\right)\right),$$

and $\alpha_{im}{}^{sd}$ can be expressed as

$$\alpha_{im}{}^{sd} = \left(t\_\mathrm{lcm}_{id}/t\_path_{id}{}^{B}\right)/\left(\left(t\_\mathrm{lcm}_{id}/t\_path_{id}{}^{A}\right) + \left(t\_\mathrm{lcm}_{id}/t\_path_{id}{}^{B}\right)\right).$$

Similarly, for split point s_s, the split ratio $\alpha_{si}{}^{sd}$ is inversely proportional to the average latency of paths A and B, and $\alpha_{sk}{}^{sd}$ is inversely proportional to the latency of path C.

4.3 | Inter-DC-pair fairness

4.3.1 | Dynamic path adjustment

To achieve inter-DC-pair fairness, DFSM allocates available paths to each unsatisfied DC pair during a path adjustment process, until all DC pairs are satisfied or nothing can be improved (no available paths remain). The process flow of dynamic path adjustment is illustrated in Figure 3. During an adjustment process, DFSM iteratively picks the least-satisfaction DC pair P among all unsatisfied DC pairs and attempts to allocate the least-latency path to P. Note that an attempted path addition for a DC pair will be confirmed only when the predicted satisfaction degree of the DC pair increases and the standard deviation (SD) of the predicted satisfaction degrees of all DC pairs decreases.

To evaluate the satisfaction degrees of all DC pairs caused by each path allocation, DFSM estimates link latencies by a prediction procedure for performance assurance.

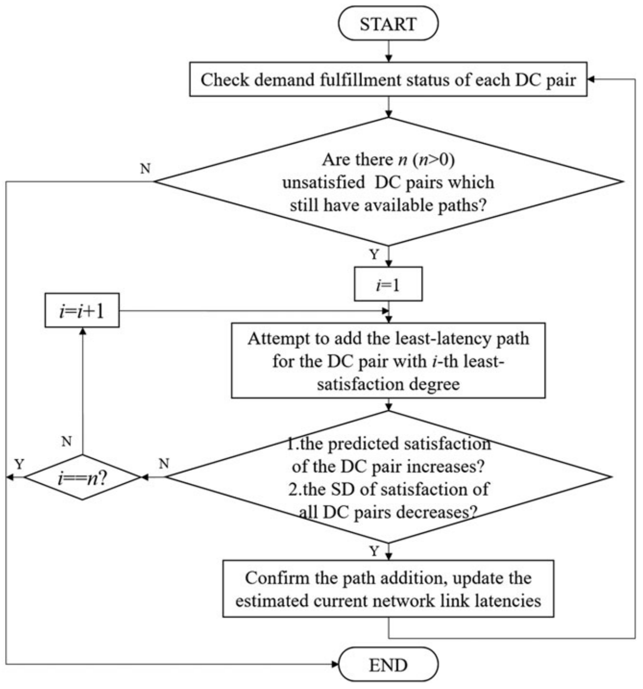
START
Check demand fulfillment status of each DC pair
Are there n (n>0) unsatisfied DC pairs which still have available paths?
N
Y
i=1
i=i+1
Attempt to add the least-latency path for the DC pair with i-th least-satisfaction degree
1.the predicted satisfaction of the DC pair increases?
2.the SD of satisfaction of all DC pairs decreases?
N
Y
i==n?
N
Y
Confirm the path addition, update the estimated current network link latencies
END

FIGURE 3 Flowchart for the dynamic path adjustment. DC indicates data center; SD, standard deviation

4.3.2 | Performance assurance

In order to verify whether an attempted path addition (or removal) for a DC pair increases (or decreases) its satisfaction and increases the inter-DC-pair fairness, DFSM introduces a performance prediction procedure to decide whether the attempted path adjustment should be confirmed.

For each path allocation to a DC pair, the prediction procedure of DFSM recomputes the ideal distribution of the ingress load of the DC pair (ie, the total traffic load sent from one DC to another). In order to reallocate the traffic load of a given DC pair, eg, *P*, we roll back to an environment with all the traffic loads of existing DC pairs (except the given DC pair *P*), and we reassign the traffic load of *P* to paths for it on the basis of intra-DC-pair fairness. The prediction procedure then computes each link latency with the estimated link load and given link bandwidth, by M/M/1 queuing model. With the derived network link latencies, DFSM is able to compute the average path latency for each DC pair.

For example, in Figure 4, the measured ingress load of the DC pair (s_s, s_d) is 40 units, and a new path is about to be added for this DC pair. Figure 4A shows the removal of traffic load of this DC pair. For example, at $link_{si}$, since the total split ratio is equal to 50% ($\alpha_{si}{}^{sd}$), the 20 units (40 * 50%) load distribution should be removed from the current load of

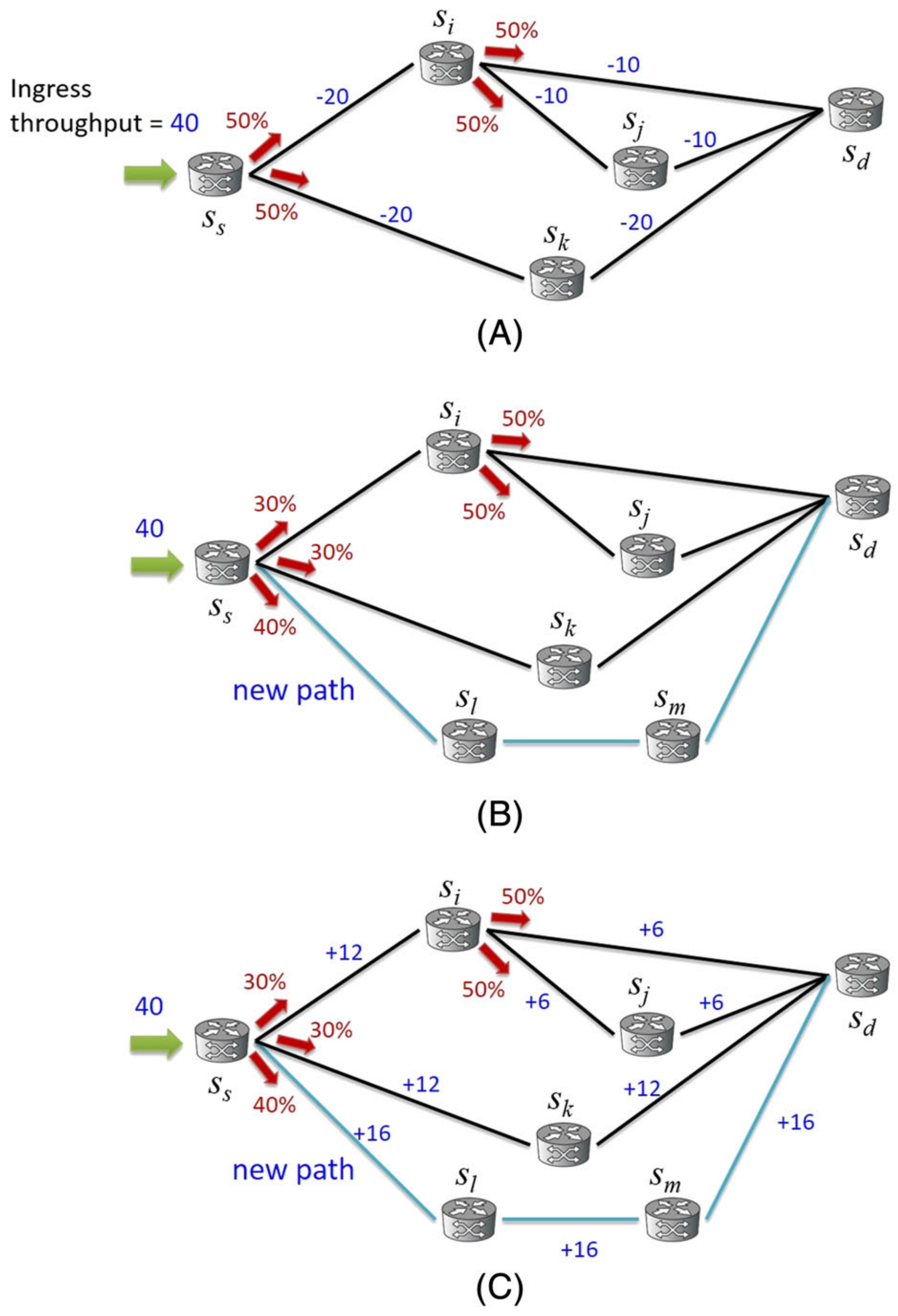

FIGURE 4 An example of link load estimation after a path addition for a given DC pair. A, Rollback to an environment without the traffic loads of the given DC pair. B, Update the split ratios after the path addition for the given DC pair. C, Reassign the traffic load of the given DC pair to paths for it. DC indicates data center

$link_{si}$. Since the total split ratio is equal to 25% ($\alpha_{si}{}^{sd} * \alpha_{id}{}^{sd}$) at $link_{id}$, the 10 units (40 * 25%) load distribution should be removed from the current load of $link_{id}$. This is similar at the other links. Figure 4B clearly shows the split ratio determined after path addition. For ease of illustration, we suppose that $\alpha_{si}{}^{sd}$, $\alpha_{sk}{}^{sd}$, and $\alpha_{sl}{}^{sd}$ are equal to 30%, 30%, and 40%, respectively, as the decision result. Since the total split ratio for $link_{sl}$ in Figure 4C is equal to 40% ($\alpha_{sl}{}^{sd}$), the 16 units (40 * 40%) load distribution should be added to $link_{sl}$. It is the same at the other links.

5 | ANALYSIS OF FLOW ENTRY SAVING

Table 3 shows the additional notations for ease of comparison between source-destination–based forwarding and label-switched tunneling.

Recall that all possible routing paths can be maintained in a group entry at each split point. For source-destination–based forwarding, *group entry*$_{sd}$ is equal to the number of split points for a pair (s_s, s_d), ie, *SP*. Since the flow entry is shared on split points and the destination node s_d, *flow entry*$_{sd}$ is equal to 1 shared flow entry on the destination node s_d + the number of split points + the sum of the number of nodes in each used paths except the destination node s_d and the split points. With the additional notations, *flow entry*$_{sd}$ is given by

$$flow\ entry_{sd} = 1 + SP + \Sigma_{i=1}{}^{m}\ (n_i - sp_i - 1) = \Sigma_{i=1}{}^{m}\ n_i - m + 1 - (\Sigma_{i=1}{}^{m}\ sp_i - SP). \quad (1)$$

Since the group entry for label-switched tunneling is only needed at source node s_s for a pair (s_s, s_d), the total number of group entries *group entry*$_{sd}$ is equal to 1. In this case, *flow entry*$_{sd}$, the total number of flow entries used for a pair (s_s, s_d), is equal to 1 shared flow entry on the source node s_s + the sum of the number of nodes in each used path except the source node s_s. So *flow entry*$_{sd}$ is given by

$$flow\ entry_{sd} = 1 + \Sigma_{i=1}{}^{m}\ (n_i - 1) = \Sigma_{i=1}{}^{m}\ n_i - m + 1. \quad (2)$$

From Equations 1 and 2, the difference of the number of system flow entries is equal to ($\Sigma_{i=1}{}^{m}\ sp_i - SP$); it means that when *SP* is fixed, the higher $\Sigma_{i=1}{}^{m}\ sp_i$ (the more intersection nodes between paths) saves more flow entries.

6 | EVALUATION AND DISCUSSION

In this section, we use Mininet[31] to evaluate the performance of DFSM. We respectively construct five inter-DC WAN topologies (A-E, as shown in Figure 5A-E) with virtual switches (Open vSwitch 2.5.2) and hosts (as local DCs) in a virtual machine (VM) that has 1 CPU and 2G RAM, and we run the DFSM application over the Ryu[32] controller in the same VM to manage the virtual switches via OpenFlow 1.3. Topologies A to D are the same as the practical topologies used by wECMP-d,[27] and topology E is the same with the B4[13] worldwide deployment (note that B4 abstracts each WAN site into a single node with a single edge of given capacity to each remote site).

For topologies A to E, each link is bidirectional, and the link capacity of each topology is 10 Gbps. For topologies A to D, six numbered edge switches are chosen to connect six DCs; and for topology E, all 12 nodes are chosen to connect 12 DCs. Each DC sends 20 TCP flows to every other DC at random rate (1 to 3 Gbps) via these selected switches. Since we have six DCs and 12 DCs in different topology, there are a total of 30 and 132 DC pairs, respectively.

TABLE 3 Additional notations

Notation	Description
m	The number of paths used for a pair (s_s, s_d)
$path_{sdi}$	a used simple path from s_s to s_d ($m \geqq i \geqq 1$)
n_i	The number of nodes in $path_{sdi}$
SP	The number of split points for a pair (s_s, s_d)
sp_i	The number of split points in $path_{sdi}$ ($SP \geqq sp_i$)
group entry$_{sd}$	The total number of group entries used for a pair (s_s, s_d)

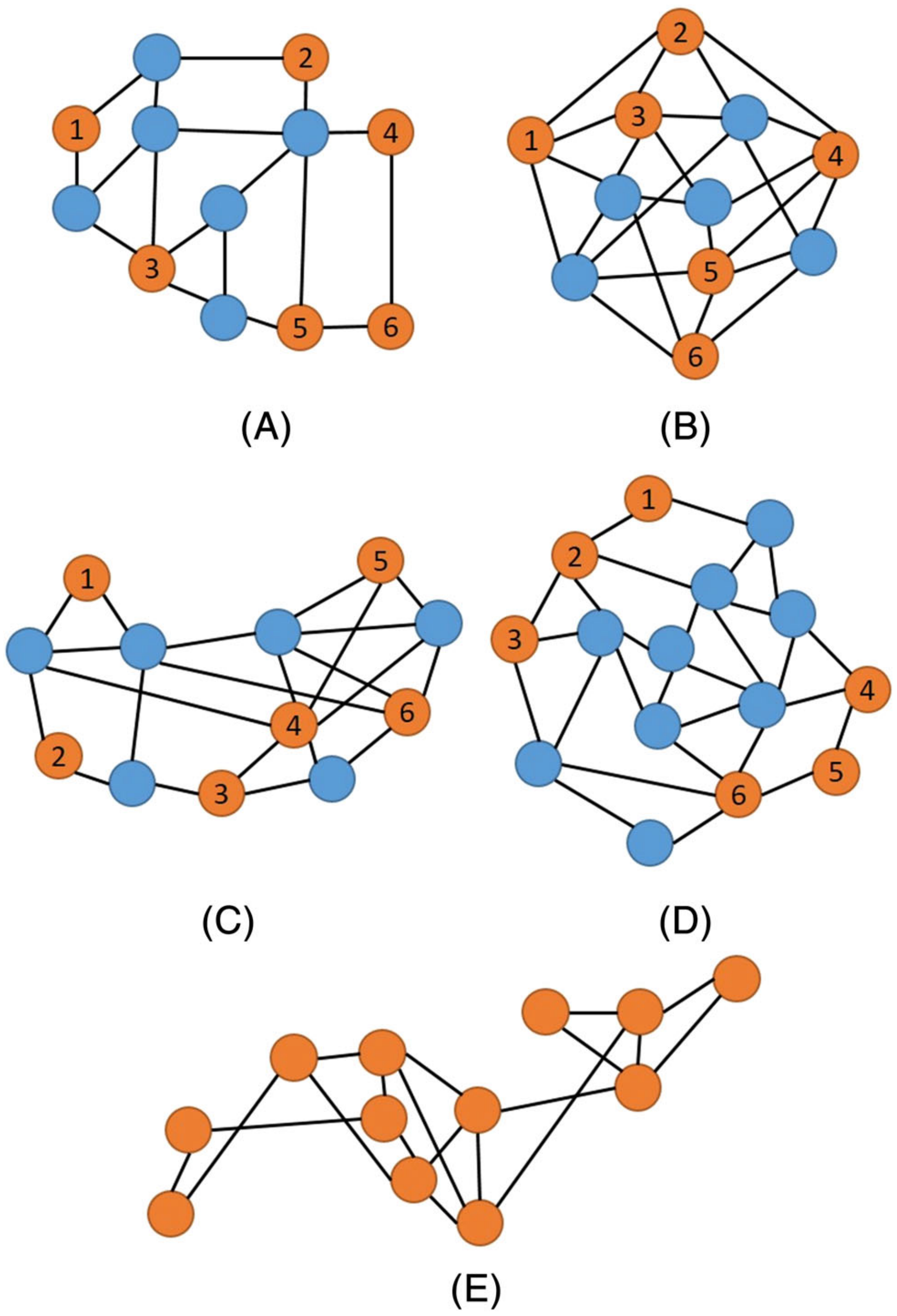

FIGURE 5 A–E, Five practical topologies with 30 to 132 DC pairs. DC indicates data center

In our evaluation, DFSM will try to satisfy the latency demand of each DC pair in each topology. In order to make DFSM's best efforts to satisfy each DC pair, both the upper bound and lower bound of latency demand set for each DC pair are equal to 0 seconds per bit so as to never be satisfied.

Table 4 gives the number of iterations and the computation time taken by DFSM. In each iteration of the path adjustment process, a path addition is performed for a selected DC pair. DFSM takes 3 seconds on average to optimize 30 DC pairs and takes 12 seconds to optimize 132 DC pairs. Since not all DC pairs need to be optimized in practice (not all DCI services are latency sensitive), the computation time could be reduced.

We evaluate DFSM in terms of the average path latency of all DC pairs and the consumption of system flow entries.

TABLE 4 The number of iterations and the computation time taken by DFSM to satisfy DC pairs

Topology	The Number of DC Pairs	The Number of Iterations	Computation Time, s
A	30	13	4
B	30	13	2
C	30	18	2
D	30	22	4
E	132	50	12

Abbreviations: DC, data center; DFSM, dynamic flow-entry-saving multipath.

6.1 | Compared with ECMP, DFSM reduces 10% to 48% latency by consuming 6% to 20% more flow entries

Figures 6 and 7 show the number of system flow entries and the corresponding average latency of all DC pairs in each topology, along with the comparison with ECMP. The results indicate that adopting k ($k = 5$) shortest paths (DFSM) reduces about 48%, 14%, 10%, and 27% of average latency of all DC pairs results from adopting ECMP in topologies A, B, C, and E, by consuming about 8%, 12%, 20%, and 6% more flow entries, respectively.

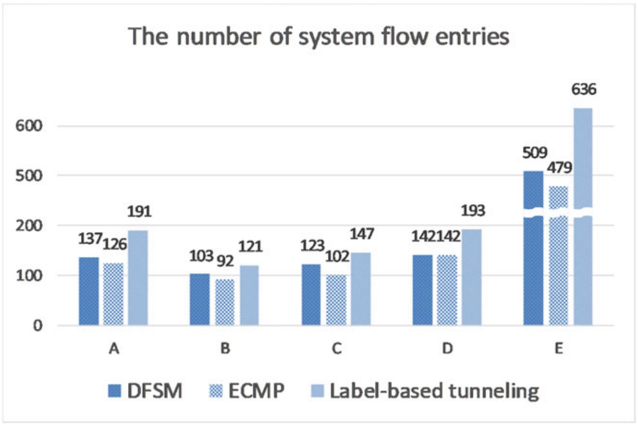

FIGURE 6 The number of system flow entries. ECMP consumed fewer flow entries than did DFSM because of the fewer available paths. Label-based tunneling consumed more flow entries than did DFSM with the same path selection because of the per-tunnel setting on intersection-forwarding nodes. DFSM indicates dynamic flow-entry-saving multipath; ECMP, equal-cost multipath

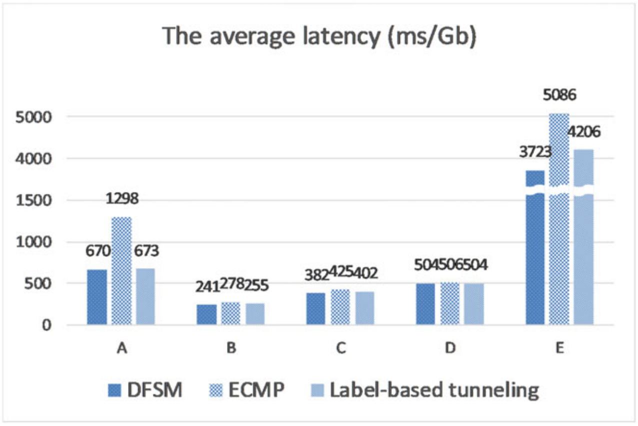

FIGURE 7 The average latency. Compared with ECMP, DFSM provides higher investment efficiency of flow entries in the less-interconnected topologies (A-E). For label-based tunneling, the average latency is similar to that of DFSM with the same path selection in most cases except in topology with more DC pairs (E). DFSM indicates dynamic flow-entry-saving multipath; ECMP, equal-cost multipath

It can be seen that DFSM provides higher investment efficiency of flow entries in topology A, since there are few equal-cost shortest paths between most pairs. Furthermore, since the nodes are highly interconnected in topology D, there are enough equal-cost shortest paths, and it makes not much difference in average latency and system flow entry number between these two cases. In general, a large-scale deployment may not have a highly interconnected topology as a result of the consideration of deployment cost and the long distance between forwarding nodes.

6.2 | Compared with label-switched tunneling, DFSM saves 15% to 30% flow entries and reduces the standard deviation of path latencies from 10% to 7%

We also applied label-switched tunneling to the same set of selected paths used in DFSM. Label-switched tunneling also splits traffic on the basis of latency estimates but only at source node. Figures 6 and 7 also show the number of system flow entries and the average latency along with the comparison with label-switched tunneling; the results indicate that DFSM saves about 30% system flow entries in topologies A and D, about 15% system flow entries in topologies B and C, and about 20% system flow entries with 12% latency reduction in topology E.

The average σ_{sd} (standard deviation of path latencies of each DC pair) is about 7% and 10%, respectively, of average path latency for all topologies when the source-destination–based forwarding and the label-switched tunneling is adopted.

Table 5 further shows the factors that affect flow entry saving. As mentioned in Section 5, for a DC pair, the difference of system flow entry number between source-destination–based forwarding and label-switched tunneling is equal to ($\Sigma_{i=1}{}^{m}\ sp_i - SP$), where SP is the number of split points for a pair, m is the number of used paths, and $\Sigma_{i=1}{}^{m}\ sp_i$ is the sum of the number of split points in each used path. In Table 5, we use the term "*mp-pairs*" to refer to pairs that use more than one path. Note that there are 27% to 57% *mp-pairs* in topologies A to E, and each *mp-pair* uses 2.2 to 2.6 paths on average in different topologies. To illustrate the flow entry saving, we take topology E for example, where each *mp-pair* saves about 3.5 (6.4-2.9) flow entries on average, compared with label-switched tunneling. Since there are 36 *mp-pairs*, the total amount of saved flow entry is equal to 126 (36 * 3.5).

6.3 | Compared with per-flow provision, per-flow provision for 600 flows (20 flows per DC pair) consumes about 16 times of flow entries

To implement per-flow provision, we assign single shortest (least-latency) path to each flow of total 600 TCP flows (20 flows per DC pair). The average latency of each pair is the average path latency of its 20 flows. Figure 8 shows the number of system flow entries and the corresponding average latency of all DC pairs in each topology (A to D), along with the comparison with per-flow provision. The result indicates that adopting per-flow provision consumes about 16 times the system flow entries consumed by DFSM but reduces latency by 10% at most (in topology D).

6.4 | Compared with even traffic splitting, DFSM reduces the standard deviation of path latencies from 14% to 7%

The average σ_{sd} (standard deviation of path latencies of each DC pair) is about 7% and 14%, respectively, of average path latency for all topologies when the latency-aware traffic splitting and the even traffic splitting are adopted.

TABLE 5 The factors that affect flow entry saving

Topology	The Number of *mp-pairs*	The Average Number of Used Paths (m) of all *mp-pairs*	The Average Number of $\Sigma_{i=1}{}^{m}\ sp_i$ of All *mp-pairs*	The Average Number of Split Points (SP) of all *mp-pairs*
A	17	2.2	5.4	2.5
B	11	2.6	3.3	1.7
C	14	2.2	3.4	1.6
D	16	2.5	5.4	2.4
E	36	2.3	6.4	2.9

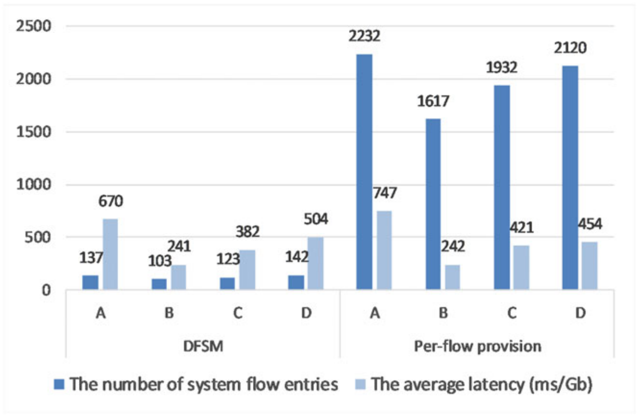

FIGURE 8 DFSM vs per-flow provision. DFSM indicates dynamic flow-entry-saving multipath

7 | CONCLUSION AND FUTURE WORK

In this paper, we propose an SDN-based TE method called DFSM for inter-DC traffic forwarding. For flow entry saving, DFSM uses source-destination–based forwarding instead of label-switched tunneling, so as to merge flow entries at intersection-forwarding nodes. Under the source-destination–based forwarding, DFSM adopts latency-aware traffic splitting at each intersection-forwarding node to achieve better flow latency control. In addition, the fairness of path assignment is also considered. Evaluation results show that DFSM, compared with label-switched tunneling, saves 15% to 30% system flow entries in practical topologies and reduces the standard deviation of path latencies from 10% to 7%. DFSM also reduces average latency by 10% to 48% by consuming 6% to 20% more flow entries than do ECMP in less-interconnected topologies. From the comparison with ECMP, it can be seen that the performance gain may not always be proportional to flow entry investment, with the interconnectivity between nodes being an important factor. The evaluation also indicates that per-flow provision consumes several times the flow entries consumed by DFSM but reduces latency by 10% at most. Besides, DFSM, compared with even traffic splitting, reduces the standard deviation of path latencies from 14% to 7%.

Since asynchronous changes of split ratios can cause transient congestion,[33] a network update usually must be performed in multiple well-planned steps. zUpdate[33] performs congestion-free DC network (DCN) updates under asynchronous switch and traffic matrix changes. Computing and implementing an update plan for inter-DC network would be much more challenging because of irregular topology. In future work, we will handle this issue.

ORCID

Yao-Chun Wang http://orcid.org/0000-0002-1531-9378

REFERENCES

1. Wang Yao-Chun, Lin Ying-Dar, Chang Guey-Yun, "SDN-based dynamic multipath forwarding for inter-data center networking," in IEEE LANMAN, Osaka, Japan, June 2017.
2. Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture. In Proc. of SIGCOMM, 2008.
3. Thaler D, C Hopps. Multipath issues in unicast and multicast next-hop selection. IETF RFC 2991, Nov. 2000.
4. Cao J, Xia R, Yang P, Guo C, Lu G, Yuan L, Zheng Y, Wu H, Xiong Y, Maltz D. Per-packet load-balanced, low-latency routing for Clos-based data center networks. In Proc. ACM CoNEXT, 2013.
5. M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In Proc. of NSDI 2010.
6. M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker. pfabric: Minimal near-optimal datacenter transport. In ACM SIGCOMM, 2013.
7. S. Fang, Y. Yu, C.H. Foh, K.M.M. Aung, A loss-free multipathing solution for data center network using software-defined networking approach, in: APMRC, 2012 Digest, IEEE, 2012, pp. 1–8.

8. Akyildiz IF, Lee A, Wang P, Luo M, Chou W. A roadmap for traffic engineering in SDN-OpenFlow networks. *Comput Netw*. Oct. 2014;71:1-30.
9. Seddiki MS, Shahbaz M, S. Donovan, S. Grover, M. Park, N. Feamster, Y.-Q. Song, FlowQoS: QoS for the rest of us, Proceedings of the third workshop on Hot topics in software defined networking, August 22-22, 2014, Chicago, Illinois, USA.
10. OpenFlow. https://www.opennetworking.org/projects/open-datapath/, August 19, 2017.
11. Kannan K, Banerjee S. Compact TCAM: flow entry compaction in TCAM for power aware SDN. In: *Distributed Computing and Networking*. Springer; 2013:439-444.
12. Blake S, Black D, Carlson M, Davies E, Wang Z, Weiss W, RFC 2475: an architecture for differentiated service (1998).
13. S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: experience with a globally-deployed software defined wan. In SIGCOMM, 2013.
14. C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer. Achieving high utilization with software-driven WAN. In Proc. ACM SIGCOMM, 2013.
15. Amin R, Shah N, Shah B, Alfandi O. Auto-configuration of ACL policy in case of topology change in hybrid SDN. *IEEE Access*. 2016;4:9437-9450.
16. Canini M, Feldmann A, Levin D, Schaffert F, Schmid S, Panopticon: incremental deployment of software-defined networking, in ACM Symposium on SDN Research. (Santa Clara, CA, USA, 2016)
17. Agarwal S, Kodialam M, Lakshman T, "Traffic engineering in software defined networks," in INFOCOM, 2013 proceedings IEEE IEEE, 2013, pp. 2211–2219.
18. Vissicchio S, Tilmans O, Vanbever L, Rexford J, "Central control over distributed routing," in Proc. ACM SIGCOMM, London, U.K., 2015, pp. 43–56.
19. Guo Y, Wang Z, Yin X, Shi X, Wu J, "Traffic engineering in SDN/OSPF hybrid network," in Proc. IEEE 22nd Int. Conf. Netw. Protocols (ICNP), Raleigh, NC, USA, 2014, pp. 563–568.
20. PICA8. https://www.pica8.com/, June 28, 2018.
21. Switch Light OS. https://www.bigswitch.com/products/switch-light, June 28, 2018.
22. Open Network Linux. http://opennetlinux.org/, June 28, 2018.
23. Cord. https://wiki.opencord.org/, August 19, 2017.
24. Hopps C, "Analysis of an equal-cost multi-path algorithm," IETF RFC 2992, November 2000.
25. Chen C, Liu C, Liu P, Loo B, Ding L: A scalable multi-datacenter layer-2 network architecture. In: 1st ACM SIGCOMM Symposium on Software Defined Networking Research, Article No. 8. ACM, New York (2015).
26. Zhang J, Xi K, Luo M, Chao H, "Load balancing for multiple traffic matrices using SDN Hybrid Routing," in Proc. 15th IEEE HPSR, Jul. 2014.
27. Zhang J, Xi K, Zhang L, Chao H, "Optimizing network performance using weighted multipath routing," in Computer Communications and Networks (ICCCN), 2012 21st International Conference on, pp. 1–7, 30 2012-aug. 2 2012.
28. Kleinrock L. *Queueing Systems, Volume I: Theory*. New York: Wiley; 1975.
29. Depth-first search and linear graph algorithms, R. *Tarjan SIAM Journal of Computing*. 1972;1(2):146-160.
30. Yen JY. Finding the K shortest loopless paths in a network. *Management Science*. 1971;17(11, Theory Series) Jul.:712-716.
31. Mininet: http://mininet.org/, August 19, 2017.
32. Ryu: https://osrg.github.io/ryu/, August 19, 2017.
33. Liu HH, Wu X, Zhang M, Yuan L, Wattenhofer R, Maltz DA, Update: updating data center networks with zero loss, Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, August 12-16, 2013, Hong Kong, China.

How to cite this article: Wang Y-C, Lin Y-D, Chang G-Y. SDN-based dynamic multipath forwarding for inter–data center networking. *Int J Commun Syst*. 2018;e3843. https://doi.org/10.1002/dac.3843