Distributed Traffic Flow Consolidation for Power Efficiency of Large-Scale Data Center Network

Kuangyu Zheng, Xiaorui Wang, *Senior Member, IEEE*, and Jia Liu, *Senior Member, IEEE*

Abstract—Power optimization for data center networks (DCNs) has recently received increasing research attention, since a DCN can account for 10 to 20 percent of the total power consumption of a data center. An effective power-saving approach for DCNs is traffic consolidation, which consolidates traffic flows onto a small set of links and switches such that unused network devices can be shut down dynamically for power savings. While this approach has shown great promise, existing solutions are mostly centralized and do not scale well for large-scale DCNs. In this article, we propose DISCO, a **DIS**tributed traffic flow **CO**nsolidation framework, with correlation analysis and delay constraints, for large-scale data center network. DISCO features two distributed traffic consolidation algorithms that provide different trade-offs (as desired by different DCN architectures) between scalability, power savings, and network performance. First, a flow-based algorithm is proposed to conduct consolidation for each flow individually, with greatly improved scalability. Second, an even more scalable switch-based algorithm is proposed to consolidate flows on each individual switch in a distributed fashion. We evaluate the DISCO algorithms both on a hardware testbed and in large-scale simulations with real DCN traces. The results show that, compared with state-of-the-art centralized solutions, DISCO can achieve nearly the same power savings while decomposing the global problem into sub-problems that are three orders of magnitude smaller. As a result, DISCO can run 10^4 to 10^6 times faster for a DCN at the scale of 10K servers. The convergence of DISCO has also been proven theoretically and examined experimentally.

Index Terms—Data center network, power savings, scalability, correlation analysis, traffic consolidation

1 Introduction

The large amount of power consumption of Internet data centers has become a serious concern in the last decade. Many recent studies have shown that there are typically three major power consumers in a data center: servers, cooling systems, and the data center network (DCN) [1], [2], [3]. As the power efficiency of data center cooling has been significantly improved in recent years (e.g., by using cold river water for cooling [4]), power consumptions by servers and the DCN are expected to be more dominant in the near future. Compared to the large body of existing research on power-efficient computer servers, power optimization for DCNs, which account for 10 to 20 percent of the total power consumption of a data center [1], [2], [3], has received increasing attention in recent years [5], [6], [7].

Among the DCN power optimization strategies, one of the most effective approaches is termed *dynamic traffic consolidation* [5], [8], which consolidates traffic flows onto a small set of links and switches, such that unused network devices can be dynamically turned off for power savings[1] and woken up later if the workload increases. Traffic consolidation is based on the key observation that DCNs are commonly provisioned for the worst-case workloads that rarely occur. As a result, a DCN can often be underutilized, leading to excessive power consumption. Similar to server consolidation [9], DCN traffic consolidation can also achieve a significant amount of power savings because the idle power (power consumption without workload) of a typical network switch is much higher than its dynamic power (power consumption corresponding to the workload) [5], [8].

While traffic consolidation has shown great promise, existing solutions are mostly *centralized* and do not scale well for large-scale DCNs. For example, ElasticTree [5] employs a centralized optimization framework, where all links and switches in the DCN are managed by a single centralized optimizer for consolidation. Likewise, CARPO [8] adopts a centralized analysis process to identify the correlation among traffic flows, such that different flows can be better consolidated if they do not peak at exactly the same time. A fundamental limitation of these centralized schemes is that their computational complexities increase dramatically with the DCN size. As shown in Table 1, the computation time of consolidation is longer than three hours for a DCN with only 10,000 servers. For DCNs with hundreds of thousands of servers, the computation time can become unacceptably long. As a result, highly scalable traffic consolidation algorithms are much needed for future DCNs whose sizes are expected to grow rapidly [7], [10].

However, there are two major challenges in designing scalable traffic consolidation schemes. First, when decomposing the global DCN power optimization problem into sets of smaller optimization sub-problems, great care should be taken for the trade-off between scalability and consolidation performances, i.e., how to efficiently decompose and design the optimizers for sub-problems such that the resulting power

1. Similar to related work [5], [6], [8], we use power and energy interchangeably here, because data center is typically required to be always on.

- *K. Zheng is with the School of Electronic and Information Engineering, Beihang University, Beijing 100083, China, and also with the Department of Electrical and Computer Engineering, Ohio State University, Columbus, OH 43210 USA. E-mail: zheng.722@osu.edu.*
- *X. Wang is with the Department of Electrical and Computer Engineering, Ohio State University, Columbus, OH 43210 USA. E-mail: wang.3596@osu.edu.*
- *J. Liu is with the Department of Computer Science, Iowa State University, Ames, IA 50011 USA. E-mail: jialiu@iastate.edu.*

Manuscript received 21 Apr. 2019; revised 7 Dec. 2019; accepted 20 Jan. 2020. Date of publication 30 Jan. 2020; date of current version 7 June 2022. (Corresponding author: Kuangyu Zheng.) Recommended for acceptance by C. Wu. Digital Object Identifier no. 10.1109/TCC.2020.2970403

TABLE 1
Computation Time[2] at Different DCN Scales

Algorithms	1,000 servers	10,000 servers	100,000 servers
ElasticTree	7.2 min	230.4 min	13,286.0 min
CARPO	8.5 min	304.3 min	15,703.3 min

savings remain close to that of a centralized scheme, while the desired scalability could be achieved. Second, there are various types of DCN architecture designs, such as fat-tree [11], VL2 [10], BCube [12], QFabric [13], and FBFLY [14]. Those different DCN architectures may require different decomposition schemes. For example, in fat-tree (an example shown in Fig. 4), which is a typical hierarchical DCN, switches are organized at three levels: core, aggregation, and edge. Hence, it is natural to decompose the global problem according to the switch levels. However, different from the switch-centric and tree-like topologies used in fat-tree, a different DCN architecture, such as BCube [12], adopts a sever-centric topology. Each server in BCube has multiple network cards for packet forwarding, thus demanding a different design of decomposition strategy. Therefore, how to design efficient decomposition schemes that are suitable for different DCN architectures is non-trivial and remains under-explored.

In this paper, we propose DISCO, a scalable power optimization framework based on traffic consolidation for large-scale DCNs. Similar to existing studies [8], [9], DISCO also leverages traffic correlation analysis to significantly reduce the DCN power consumption. However, in contrast to previous centralized solutions, DISCO features two scalable traffic consolidation algorithms that provide different trade-offs (as desired by different DCN demands) between scalability, power savings, and network performance. First, a flow-based algorithm (DISCO-F) is proposed to conduct consolidation for each flow individually for better scalability, but with slightly less power savings. Second, an even more scalable switch-based algorithm (DISCO-S) is proposed to consolidate flows on each individual switch in a distributed fashion with better scalability and power savings, at the cost of slightly longer convergence time and delay. Moreover, since the network delay performance has been identified as an important metric in DCN services [15], [16], delay constraints are included in the traffic consolidation to guarantee the network performance. Specifically, the major contributions of this paper are:

- We propose the framework of DISCO with two variants and analytically compare them against the state-of-the-art centralized solutions. Results show that DISCO can lead to a problem size that is more than three orders of magnitude smaller for each individual optimizer. The network delay constraints are incorporated into the traffic consolidation of DISCO. It significantly improves the DCN delay performance, compared to previous schemes that ignored this important metric.
- We theoretically analyze the convergence of DISCO, and prove that both DISCO algorithms converge to a stable state within a finite number of execution rounds. Experiment results also confirm that DISCO stabilizes quickly compared with the consolidation period.
- We evaluate DISCO algorithms both on a hardware testbed and in large-scale simulations with real DCN traces. Extensive experiments are conducted on the two typical DCN topologies, fat-tree and BCube, under different scenarios. Our results show that DISCO achieves nearly the same power savings and network delay as those centralized solutions, but at a much smaller communication overhead.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 provides the background of correlation analysis. Section 4 introduces the design of the proposed DISCO algorithms. Section 5 analyzes the convergence. Section 6 presents the evaluation results of DISCO algorithms in experiments with different DCN topologies and scenarios. Section 7 concludes the paper.

2 Related Work

There are several existing approaches designed for DCN power optimization [5], [8], [17], [18], [19]. One approach is link adaptation [6], [20], which dynamically adjusts the speed of each switch link according to flow bandwidth requirements to save port power. Another more efficient approach is traffic consolidation, which consolidates traffic flows onto a small set of links and switches, such that unused network devices can be dynamically shut down for power savings [5], [8], [21]. We note, however, that these existing works are centralized schemes, which require global information and do not scale well in terms of computational complexity and control overhead as the DCN size increases.

On the other hand, scalability of DCN management has also been studied, but mainly on network performance management rather than power efficiency. For example, DARD [22] and DiFS [23] propose distributed adaptive routing methods to achieve load balancing by moving elephant flows from overloaded paths to underloaded paths. EATe [24] is one of the earliest studies to use edge routers in switch-centric topologies to aggregate traffic flows in a distributed manner to save power. REsPoNse [25] is another DCN power saving method to address the scalability issue by identifying a few always-on energy-critical paths offline to save computation efforts. Different from them, both DISCO algorithms are not dependent on any specific type of DCN topology, and utilize the correlation-aware consolidation, which has been proven to provide much better power savings [8], [9].

3 Correlation-Aware Consolidation: A Primer

We first introduce the basic concept of correlation-aware traffic consolidation, which will serve as the foundation of our DISCO power optimization design. Correlation-aware consolidation has been shown to provide over 20 percent more power savings [8] than traditional method like ElasticTree [5].

Fig. 1 illustrates two traffic consolidation examples: positively correlated flows and negatively correlated flows, where the flows are normalized to the link capacity. As shown in Fig. 1a, the more positively two traces are correlated, the more likely they will have their peak/valley values appear at the same time, such that their sum (the dashed line) will have a magnified variation. In the example, the total load of the two consolidated flows exceeds the link capacity. In contrast, for two non-positively correlated traces, (e.g., f_1 and f_3 in Fig. 1b), their sum will have less variation, thus requiring less capacity for consolidation. The correlation degree can be quantified by the Pearson

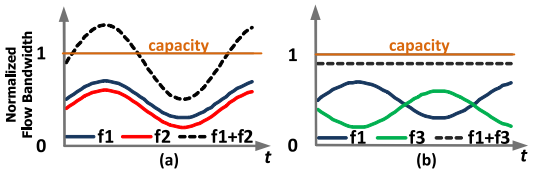


Fig. 1. Correlation-aware consolidation of two example flows leads to lower capacity requirement and so better consolidation (the dashed line is the aggregated load). (a) Positively correlated, with variation magnified in the aggregation. (b) Negatively correlated, with variation canceled in the aggregation.

Correlation Coefficient [8] between each flow pair by

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}, \quad (1)$$

where x_i and y_i are the bandwidth demands of flows x and y at each sample point within one consolidation period.

In a correlation-aware consolidation approach, different DCN flows are consolidated based on their non-peak values and correlation relationship. This is based on the observations that: 1) most of the time, the load of a DCN flow is much lower than its peak value [26], [27], and 2) most of the loads of different traffic flows are weakly correlated, and hence do not always peak at the same time [8], [18].

Based on these two observations, a correlation-aware solution consolidates different traffic flows using their non-peak workloads, under the constraint that the correlations between these traffic flows are below a certain threshold. This approach has been demonstrated to yield more power savings [8], [9]. Different from previous work that focus mostly on centralized correlation-aware traffic or server consolidation, our DISCO algorithms decompose correlation analysis into localized power optimizers for scalable distributed traffic consolidation in large-scale DCNs.

4 Design of DISCO

In this section, we present our proposed DISCO optimization framework, which includes two scalable traffic consolidation algorithms, namely DISCO-F and DISCO-S.

4.1 The DISCO Framework

Since the traffic consolidation problem is known to be NP-hard [8], [28], we propose a feasible algorithmic framework DISCO featuring two different algorithms, with performance guarantee. Before diving into details, we first provide an overview for the two proposed algorithms with the framework illustrated in Fig. 2. Both DISCO algorithms share the same two general processes: 1) Correlation Analysis. DISCO-F uses the optimizer on each flow source host server; DISCO-S uses the optimizer on each switch. 2) Traffic Consolidation. The local optimizers (flow-host-based ones for DISCO-F; switch-based ones for DISCO-S) in a distributed way, determine the paths of related flows, conduct adjustment when monitoring congestion, then dynamically turn on\off devices for power savings. These two processes are performed periodically to adapt to the traffic variations (details are in the designs of each algorithm below).

Incorporating Delay Constraints. DISCO incorporates the delay constraints during the traffic consolidation based on the DCTCP protocol [29]. DCTCP leverages the Explicit Congestion Notification (ECN) function by setting a clear

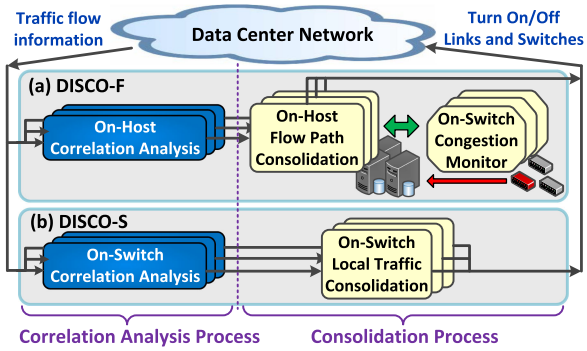


Fig. 2. The DISCO framework includes two scalable consolidation algorithms that are designed for different DCN architectures and can be selected based on the different trade-offs between scalability, power savings, and delay performance: (a) the flow-based DISCO-F; (b) the switch-based DISCO-S.

queuing packet number notification threshold K to provide more fair bandwidth sharing and less queuing delay. It is proved that the steady-state queue-length will be at most as $Q_{MAX} = K + N_s * W_s$ [29], [30], where N_s is the number of flows, $W_s = (C * RTT + K)/N_s$ is the window size of the short flow in one round trip time (RTT) under link capacity C. Then, following the same model in [31], the total flow completion time (FCT) for a flow with L_s packets can be estimated as $FCT = (L_s/W_s) * (RTT + D_q + \sum_{i=1}^{n} D_i)$, where $D_q = Q_{MAX}/C$ is the maximum queuing delay; D_i is the packet processing delay of switch i along the flow path; n is the number of switches on the path. Therefore, during the traffic consolidation, DISCO can estimate the FCT_j for a flow f_j based on the existing flows and the number of switch on a candidate path, and only consider the paths satisfying the delay constraint $FCT_j \leq D_{req}$, where D_{req} is the delay requirement. In what follows, depending on the DCN architecture, we discuss two decentralized schemes.

Flow-Based DISCO. To overcome the limitations of centralized algorithms and design a practical decomposition strategy, we first identify the general common features of DCN structures. For all DCN architectures including both hierarchical ones or non-hierarchical ones, since all flows in a DCN start from a server, it is intuitive to decompose the consolidation problem at the flow level. One feasible method is let each source server manage the paths of the flows starting from it. To be specific, each flow can have an optimizer on its source server to conduct consolidation only for this flow, such that the problem size is reduced to the length of the flow path (i.e., the number of switches this flow passes through). Hence, each optimizer can make decision independently with a much reduced problem size, which is more scalable and does not depend on any particular type of DCN topology. We name this algorithm DISCO-F.

As a distributed algorithm, DISCO-F also has some limitations: Due to the lack of global information, the consolidation decision of each flow is local-optimal in general. As a result, DISCO-F may provide less power savings than the centralized solutions. On the other hand, since each optimizer still needs the knowledge of all the switches on the flow path, the computational problem size dependents on the length of the path. This problem size could still be high in a large-scale DCN that has many long flows.

Switch-Based DISCO. To further reduce the problem size and achieve better power savings, we propose an even

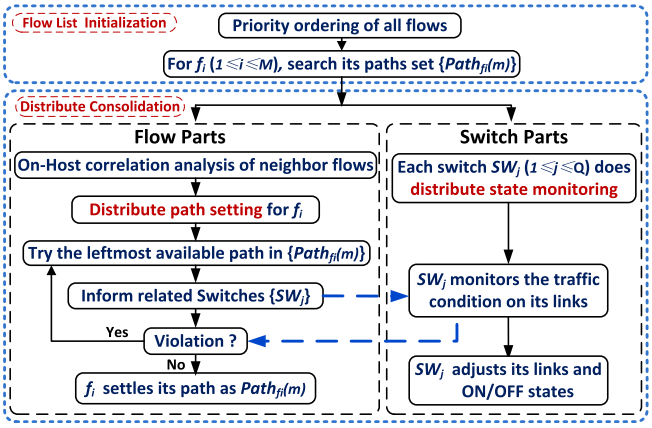


Fig. 3. The algorithm flowchart of algorithm DISCO-F.

more scalable switch-based algorithm DISCO-S. In this algorithm, each switch has a distributed optimizer running on its processor to consolidate only the flows that pass through. Therefore, in DISCO-S, the problem size of each optimizer is even lower than that of DISCO-F. Moreover, we design DISCO-S to conduct traffic consolidation aggressively, for more possible power savings. In addition, DISCO-S is also not dependent on any particular type of DCN structures.

DISCO Implementation. To implement DISCO, for DISCO-F, the optimizer of each flow runs on its source server. For DISCO-S, the optimizer runs on the main processor of each switch. Moreover, the current software defined network (SDN) switches (e.g., Open-Flow [32] based switch) already offer the traffic forwarding table management with remote on-server controller, which can control groups of switches at the same time. Note that even though most of current Open-Flow systems work in a centralized scheme, we propose to utilize Open-Flow technology distributively to address the scalability issue specifically for the DCN power optimization problem. Meanwhile, there are already some systems developed based on Open-Flow in a distributed manner by having multiple controllers for different switch groups [33], [34], which shows the feasibility of this direction. In addition, the Open-Flow switch can also provide functions for per-flow/per-port statistics, which enable the large-scale implementation of the DISCO algorithms. Similar to previous work [5], [8], DISCO uses the traffic load information in the previous period as the estimation for current period. But even when unpredictable traffic occurs, DISCO is designed to have follow-up path adjustments. In addition, prediction methods [5] can be integrated with DISCO, but is not the focus of this paper.

4.2 DISCO-F: Flow-Based Algorithm

DISCO-F includes two parts as shown in Fig. 3.

1) Initial Flow Consolidation. In DISCO-F, each flow optimizer begin with the search of the available flow path sets for each f_i, denoted as $\{Path_{f_i}(m)\}$ $(1 \leqslant m \leqslant P_i)$, where P_i is the number of available paths for flow f_i. In each period, the flow-level optimizer shares its flow bandwidth requirements with other optimizers in the same pod, and conducts the local correlation analysis of the *neighboring flows*. Here, we define neighboring flows as the flows originating from the same server or the other servers in the same pod, who have higher chances to share the same links. Based on this analysis, each optimizer knows the correlations among the neighboring flows and the non-peak (e.g., 90-percentile) bandwidth requirement of each f_i. To avoid the redundant computations within the neighborhood, this correlation analysis can be done on one single optimizer (e.g., the leftmost one in the neighborhood), then send the result to other neighbor optimizers. Then, each optimizer tries to assign f_i to an available path in $\{Path_{f_i}(m)\}$ according to the lexicographic order (i.e., from the path with smaller index in a general DCN topology) to save power. Here we define the *consolidation constraints* as follows: 1) the bandwidth requirements of f_i should be smaller than the remaining capacity of the candidate link, 2) the correlation coefficient between f_i and any flow on the candidate link should be lower than the threshold, and 3) the estimated delay should be smaller than the delay constraint. If there is any constraint violation, the optimizer will try the next path in order. However, since different optimizers choose paths independently, there may be transient network congestion on some links. Therefore, these congested path settings need to be adjusted in the second step.

Note that, to avoid the impact of flow workload estimation error due to DCN traffic burst, similar to previous work [5], [8], during consolidation, DISCO can reserve a link bandwidth margin (e.g., 2 percent) as redundancy for error tolerance. Meanwhile, there are also many studies on DCN traffic prediction based on machine learning algorithms [35], [36] (e.g., neural network, sequential pattern mining). However, we focus on the distributed design of DISCO in this paper, as prediction is not our focus here. In addition, during the correlation analysis, redundant computations within the neighborhood may exist, which is a trade-off for the distributed design. Further improvement scheme like computation on one single node then sharing with neighbors will be considered in our future work.

We assume there is a priority order of the DCN traffic flows, which can be based on different service types, importance, or bandwidth requirements. This priority order will be used in the processes of path adjustment.

2) Flow Path Adjustment. For all Q switches, each switch SW_j inspects the rate of each passing flow and the utilization of its links. When congestion occurs, SW_j identifies related flows starting from the lowest-priority to resolve the congestion condition, by notifying the corresponding flow optimizers for path adjustments. This process keeps running until the congestion is resolved or all options are tried. The convergence of this process will be proved in Section 5.

After all flow paths are settled, each unused switch puts itself to sleep until the beginning of the next period. In the implementation, this decision is made after the global convergence time (details in Section 5).

Example. Fig. 4 shows a 4-pod fat-tree with four flows (without loss of generality, we assume that f_1 to f_4 are of decreasing priority). It illustrates an example that uses two periods to converge to a stable state that has no congestion. First, the flow optimizers on the leftmost source servers in pod (i.e., S_1, S_5) perform local correlation analysis among neighboring flows, and calculates the percentile bandwidth requirement value of each flow (normalized percentile bandwidth are marked in the parentheses in Fig. 4). For example, for f_1, only f_2 is its neighboring flow. So, S_1 will only calculate the correlation coefficient between f_1 and f_2. Then, it sends the result to S_3. Similarly, S_5 will perform correlation analysis between the neighboring flows f_3 and f_4, and send result to S_7. In this example, only f_1 and f_4 are

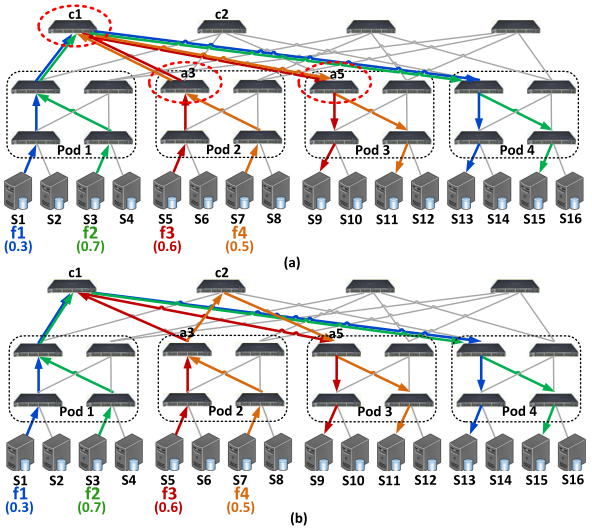


Fig. 4. Example of DISCO-F on a 4-pod fat-tree topology. (a) First step: Each flow is individually consolidated to the leftmost side of the network for power-saving in a distributed way. Congestion occurs on the dash-circled switches. (b) Second step: with the congestion notification from switches c_1, a_3 and a_5, flow f_4 (with a lower priority) adjusts its path from c_1 to c_2 to relieve the congestion.

positively correlated, which violates the correlation threshold. Then, all flow optimizers begin the distributed consolidation by choosing the available leftmost paths for their flows. However, the result (Fig. 4a) shows congestions occur on switches c_1, a_3 and a_5 (in red-dashed circle) due to the lack of knowledge of other flow paths. Then, the congested switches exam the corresponding flows (f_1 to f_4 here) and find that f_4 has the lowest priority. Thus, the congestion can be resolved by removing f_4 from the congested links. In the second step (Fig. 4b), the optimizer of f_4 on S_7 updates its path to the next leftmost path, which resolves the congestion.

4.3 DISCO-S: Switch-Based Algorithm

DISCO-S is also a fully distributed algorithm, in which every switch performs traffic consolidation individually. As shown in Fig. 6, the switch optimizer on SW_j starts the correlation analysis only among the flows that pass itself. Since there are usually multiple available forwarding links for each flow, SW_j performs traffic consolidation by choosing links in the lexicographic order. When all passing flows f_{ij} are settled, SW_j will be put to sleep by its optimizer if it is unused. Similar to DISCO-F, in the implementation, this decision is made after the global convergence time (Section 5). Note that, due to the local correlation analysis and limited path information, each switch can only choose the new flow path according to its own current knowledge. Thus, switches may aggressively consolidate flows to some shared paths, which leads to fewer usage of switches/links, but causes new congestion and requires further adjustments. Hence, DISCO-S may use fewer devices with more power savings, but result in larger network delay and longer adjustment time.

Example. Fig. 5 shows an example that uses three steps to converge to a stable state that has no congestion. First, all switches try to use the leftmost links for their flows. This aggressive step causes congestion on all the switches with flows (in dashed red circle) in Fig. 5a.

In the second step (Fig. 5b), each switch finds the congested links with the related flows, then starts to change the paths from the flow with the lowest priority. For switch e_1, the congested link between e_1 and a_1 involves flows f_1 and f_2. Since f_2 has a lower priority, e_1 only forwards f_2 to a_2. Similarly, for switch e_2, congestion involves f_3 and f_4. So e_2 updates the path of f_4 (lower priority) to a_2.

Note that the adjustments could lead to new congestion due to the distributed nature. For example, when e_1 forwards f_2 to the a_2, and e_2 also forwards f_4 to a_2, new congestion occurs on switch a_2, c_1 and a_4 in Fig. 5b.

In the third step, switches run another round of path adjustments. Similar to the previous step, the congested switch a_2 compares the priority of related flows f_2 and f_4. Since f_4 has a lower priority, a_2 changes its forward path from c_1 to c_2. The final paths of all flows are shown in Fig. 5c.

4.4 DISCO With BCube Topology

BCube [12] is another popular type of DCN structures that has a hierarchical architecture constructed with commodity switches. Different from the switch-centric tree-like topologies that are used in the previous section (e.g., fat-tree), BCube is a sever-centric topology. Each server in BCube has multiple network cards for packet forwarding. According to the definition of BCube, as a recursive topology, a level-k BCube structure unit, $BCube_k$ ($k \geq 1$), is constructed from n level-$(k-1)$ $BCube_{k-1}$ blocks and n^k n-port switches. Each server in a $BCube_k$ has $k+1$ ports. Thus, each $BCube_k$ has $N = n^{k+1}$ servers and $(k+1)n^k$ switches in total. Fig. 14 illustrates an example of BCube topology with $k = 1, n = 4$, in which a $BCube_1$ topology is constructed with 4 $BCube_0$ and 4 4-port switches, with in total 16 servers.

The BCube topology has many advantages, such as convenient to scale-up for large-size DCNs, short server-

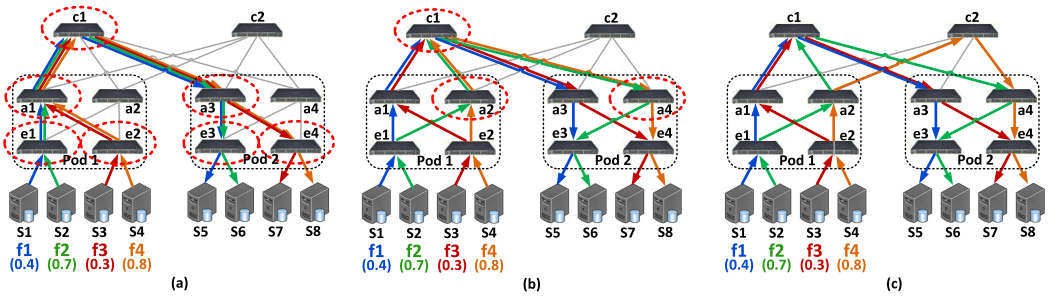


Fig. 5. Example of DISCO-S on a 2-pod fat-tree topology. (a) First step: all switches try to use the leftmost side of the topology distributively. Congestion occurs on some switches (in red dash circle). (b) Second step: congested switches do path adjustments beginning from flows with lower priorities. Paths of f_2 and f_4 are adjusted. However new congestions occur on switches a_2, c_1 and a_4. (c) Third step: because f_2 and f_4 cause congestion, switches a_2, c_1 and a_4 adjust the path of f_4 (with a lower priority). Then the congestion is relieved.

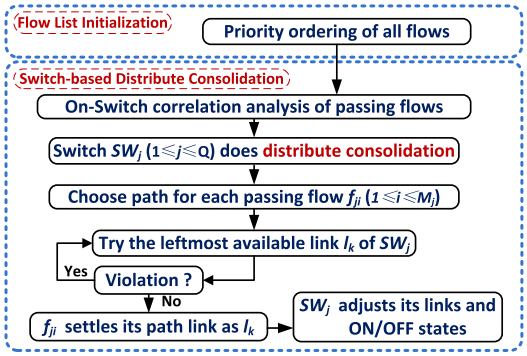


Fig. 6. The algorithm flowchart of the algorithm DISCO-S.

to-server network distance (i.e., hop number) and providing multiple parallel paths between every pair of servers. Specifically, for a level-k $BCube_k$, there will always be $k+1$ parallel paths between any two servers. This feature of path redundancy provides DISCO algorithms the chance to do traffic consolidation, and turn off unused switches/links for power savings. Note that, because of the recursive nature of BCube that is different from fat-tree which have the same fixed length for the multiple shortest path between every pair of servers, the $k+1$ parallel paths between servers have different lengths (hop number). Therefore, for each flow in the path ordering phase of DISCO algorithms, we order the sequence of its paths first from the shortest to the longest length, then use the lexicographic order for the paths with the same length. In general, DISCO algorithms still try to consolidate flows to the path with a shorter length first for possible less delay. We further test the DISCO algorithms with BCube in the evaluation in Section 6.5.

5 Theoretical Analysis

In this section, we first analyze the size of solution search space of the two DISCO algorithms. Then, we prove the convergence of DISCO algorithms.

5.1 Search Space and Communication Overhead

The size of solution search space of each optimizer directly determines the time complexity of an algorithm and impacts its scalability [33], [34]. Meanwhile, the communication overhead incurred by the optimizers (e.g., to collect and exchange information) is also an importance factor for scalability [22], [23]. Thus, we analyze and compare these two metrics of a single optimizer used in each algorithm.

We define the following notation under the fat-tree topology: k is the scale degree of a fat-tree; N is the number of servers; Q is the number of switches; M is the number of flows; M_{Pmax} is the max number of flows in a pod.

Solution Search Space Analysis. Consider a k-pod fat-tree DCN with Q=5k^2/4 switches and N=k^3/4 servers. Due to space limitation, we assume a network setup as follows for simplicity: 1) there is only one traffic flow between a pair of servers, and 2) each server only connects with one flow. Then, the total flow number M is N/2=k^3/8. Note that for general DCNs with multiple flows between a pair of servers, by simply adjusting the value of M accordingly, the analysis still holds.

Centralized CARPO and ElasticTree determine the paths of all the M flows. For each flow, they need to consider both $k/2$ aggregation switches in the source/destination pods, and the $k^2/4$ core switches, which leads to a combination of up to $k^4/16$ conditions. Thus, the solution search space is $O(Mk^4) = O(k^7)$. In Hier-CA, for each flow, the core-level optimizer only determines the core switches to use, with a solution search space of $O(Mk^2)$. The pod level optimizer determines the local flow paths with a solution search space of $O(M_{Pmax}k/2)$ ($M_{Pmax} \leqslant k^2/4$ in the example case). Therefore, the solution search space of Hier-CA is $O(Mk^2)$ + $O(M_{Pmax}k/2) = O(k^5)$. In DISCO-F, each flow-based optimizer searches all possible switch conditions but only for one flow. So, its solution search space is $O(k^4)$. In DISCO-S, the solution search space of each switch-level optimizer depends on the number of passing flows, which is $O(M) = O(k^3)$.

TABLE 2
Size of Solution Search Space

	CARPO / ElasticTree	Hier-CA	DISCO-F	DISCO-S
General Case	$O(Mk^4)$	$O(Mk^2 + M_{Pmax}k)$	$O(k^4)$	$O(M)$
Simple Case	$O(k^7)$	$O(k^5)$	$O(k^4)$	$O(k^3)$

Table 2 summarizes the sizes of solution search space of the algorithms. Compared with CARPO, DISCO-F successfully reduces the search space by *at least three orders of magnitude* of k. As shown with the *log scale* in Fig. 16a, the difference can lead to as much as $\mathbf{10^4}$ to $\mathbf{10^5}$ times less computation overhead for a DCN at the scale of 10,000 servers. DISCO-S further reduce this overhead to $\mathbf{10^6}$ times less.

5.2 Convergence Analysis

Due to the nature of distributed design that may not provide the global optimal result directly as the centralized design, it is possible that at links for some flows, both DISCO-F and DISCO-S could have transient flow congestions, and need path adjustment for several rounds. Hence, it is important to first ensure that the DCN system can converge to a stable state, i.e., the DISCO algorithms can stop flow path adjustment within a finite number of iterations. We define the *convergence time* as the time interval from the time point when the algorithms start to make adjustments to the point when the system reaches the stable state. To facilitate the convergence analysis, we assume that the traffic workload is quasi-stationary in our time-scale of interest (i.e., the bandwidth requirement of each flow is approximately constant within each period). This is consistent with the observation made from real DCN flow traces in previous studies [8], [18], for adopting the non-peak percentile workload for the correlation analysis in Section 3.

Theoretically, we also model and prove the convergence feature of both DISCO-F and DISCO-S for general DCN topologies.

5.2.1 Notation

We use the following notation for modeling and analysis.

- $\mathcal{S}$: Set of servers, with $|\mathcal{S}| = N$.
- $\mathcal{SW}$: Set of switches, with $|\mathcal{SW}| = Q$.
- $\mathcal{L} = \{l(x, y)\}$: Set of links, with $|\mathcal{L}| = L$, where $l(x, y) \in \mathcal{L}$ represents that switches SW_x and SW_y are connected by link l.

- $\mathcal{F}$ =$\{f_i\}$: Set of flows, with $|\mathcal{F}| = M$. Let M_x denote the flow number passing the switch SW_x.
- $\{Path_{f_i}(m)\}$: The set of routing paths for flow f_i, $(1 \leqslant m \leqslant P_i)$, where P_i =$|\{Path_{f_i}(m)\}|$ is the number of available paths for flow f_i.
- $\{l^x_{f_i}(n)\}$: The forwarding link set for flow f_i on SW_x, $(1 \leqslant n \leqslant L_{xi})$. L_{xi} =$|\{l^x_{f_i}(n)\}|$ is the number of available forwarding links for flow f_i.
- k : the degree of the fat-tree topology, which equals to the number of pods and the number of links of each switch.
- C : the capacity of each link.
- W_i : the bandwidth requirement of flow f_i in current period. $W_i \leqslant C$, $(1 \leqslant i \leqslant M)$.

For link $l(x, y)$ with M_a passing flows, congestion occurs when $\sum_{i=1}^{M_a} W_i > C$.

In the analysis, we assume that the bandwidth requirement of each flow is quasi-stationary, i.e., the requirement is a finite constant within each period and updates from one period to the next. This is reasonable since through the correlation-aware consolidation (Section 3), all DISCO variants use a percentile value (e.g., 90 percent) of the bandwidth requirement to represent the demand of each flow in one period, which has been demonstrated to provide better power saving performance.

As mentioned in Section 4, each algorithm of DISCO begins with a priority ordering of all flows (f_i, $1 \leqslant i \leqslant M$), sorted in decreasing priorities.

Moreover, for a given DCN topology, $\{Path_{f_i}(m)\}$ is finite and determined in the initialization stage of DISCO. Further, we let paths in $Path_{f_i}(m)$ be ordered in a lexicographic fashion under the given switches' labeling. It means the path with the smaller alphabetical switch-label sequence will be ranked with a lower index. For example, in tree-like DCN topology such as the fat-tree (e.g., Fig. 4), paths are ordered from the leftmost side, where all core/aggregation/edge switches have smaller labels, to the rightmost side. For the other central symmetric topologies (e.g., DCell [7] and QFabric [13]), an arbitrarily chosen path can be named the first path with label m = 1. We then labels the remaining paths in the clockwise manner. As a result, distributed optimizers can do flow consolidation by trying to use paths with smaller indices.

Similarly, for a known DCN topology, $\{l^x_{f_i}(n)\}$ is finite, fixed and determined in the initialization stage of DISCO-S according to the source and destination host of each flow. Similar to $\{Path_{f_i}(m)\}$, the forwarding links of f_i for each switch are also labeled in a lexicographical order (e.g., from the leftmost side to the rightmost side in the fat-tree topology).

5.2.2 Analysis of DISCO-S

As mentioned in Section 4.3, when SW_x detects congestion on any of its incident link $l(x, y)$, the DISCO-S optimizer performs forwarding link adjustments. Among all the M_x passing flows $\{f_i\}$, $(1 \leqslant i \leqslant M_x)$, the optimizer removes all the f_i needed, starting from the lowest priorities, i.e., i= M_x, M_x-1, M_x-2, ..., until link $l(x, y)$ is not congested, or until all forwarding options in $\{l^x_{f_i}(n)\}$ are tested. In DISCO-S, when forwarding link needs adjustment for f_i, the optimizer uses the next available link $l^x_{f_i}(y_{new})$, y_{new}=$y_{old} + 1$, unless $l^x_{f_i}(y_{old})$ is the last available forwarding link for f_i, i.e., y_{old}=L_{xi}. Based on this process, we have the following result:

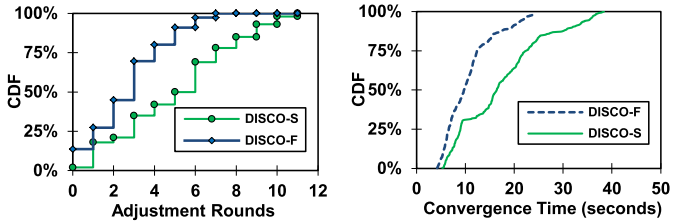


Fig. 7. CDF of (a) adjustment rounds, and (b) convergence time, for all the optimizers (512 in DISCO-F and 320 in DISCO-S) in the 1024-server simulation. Each server has 3 flows randomly connecting to 3 paired servers.

Proposition 1. *The congestion adjustments of DISCO-S terminates within a finite number of rounds.*

Proof. Consider the worst case with the largest possible amount of adjustments, where each of the switches $\{SW_x\}$, $(1 \leqslant x \leqslant Q)$, is congested by all the passing M_x flows f_i $(M_x \leqslant M)$, and each switch needs to test the largest possible rounds for all flows. Without loss of generality, we assume that flows are sorted in a decreasing order of priority.

For each switch SW_x, start from the initial state that each f_i $(1 \leqslant i \leqslant M_x)$ uses its respective first forwarding link $l^x_{f_i}(1) \in l^x_{f_i}(n)$, $(1 \leqslant n \leqslant L_{xi})$. Since only the flows with lower priorities (larger indices) will need to be adjusted, after the first round, in the worst case only f_1 will settle its forwarding link $l^x_{f_1}(1)$, and all the other $M_x - 1$ flows are adjusted to their next forwarding links. In the second round, f_2 will settle its forwarding link $l^x_{f_2}(2)$, and the remaining $M_x - 2$ flows are adjusted to their next forwarding links. Therefore, for SW_x, this process will stop after M_x rounds (when $M_x \leqslant L_{xi}$), or L_{xi} rounds (when $M_x > L_{xi}$). For convenience, we denote the upper bound number of search rounds on each switch as B = Min$\{M_x, L_{xi}\}$, $(1 \leqslant i \leqslant M_x)$. Since the workload of each flow is considered quasi-stationary as mentioned within a period, once the paths of flows with higher priorities are settled, they do not need to be changed in this period.

Hence, in the worst case, for one switch optimizer of DISCO-S, it will stop after B rounds as the upper bound. Meanwhile, similar to the analysis in DISCO-F, as each distributed optimizer reduces its own option search space in every iteration, the total global solution search space of DISCO-S is also reduced. After all optimizers terminate adjustments, which is reachable as shown above, the whole system of DISCO-S will become global stable. The global search space of DISCO-S is upper bounded by $Q \cdot B = O(Q \cdot M)$. As a special case of the above process, for the k-pod fat-tree topology, each switch has k links, so that B = Min$\{M_x, k\}$. Meanwhile, there are in total $Q = 5/4k^3$ switches. Thus, the total global search space of link options for DISCO-S is upper bounded by $O(Mk^3)$.

For the congestion adjustments of DISCO-S, there will be fewer flows to be adjusted each round, with reduced remaining path set search space. Hence, DISCO-S will terminate within a finite number of rounds. This completes the proof. □

Fig. 7a compares the Cumulative Distribution Function (CDF) of the number of rounds for convergence of each optimizer in the 1024-server simulation (each servers has 3 flows randomly connecting to another 3 paired servers). In

DISCO-F, about 90 percent of the flows finish the adjustment within 5 rounds, while in DISCO-S, it needs 9 rounds to settle 90 percent of switches. These results show that DISCO algorithms can successfully reduce the computation time, even with some more rounds of adjustments.

Convergence speed is another important performance metric for distributed algorithms. As the experiment results shown in Fig. 7b, 90 percent of the flows in DISCO-F can converge within 20 seconds. The convergence time of DISCO-S is a little longer: 90 percent of the switches finish adjustments within about 34 seconds. However, compared to the 10-minute period adopted by the correlation analysis design, the convergence time-scale of DISCO is sufficiently small.

Note that, based on the observation in correlation analysis (Section 3), the flow conditions are stable within each period, which is consistent with previous work [5], [8]. However, if the flows are shorter or longer in other scenarios, the algorithm period can be adjusted according to the specific cases, while the convergence of algorithms should still hold as proved.

6 Evaluation

In this section, we first introduce the baselines used for comparison, then evaluate DISCO in terms of power savings and network performance (i.e., packet transmission delay), both on a hardware testbed and in large-scale simulation.

6.1 Baselines Used for Performance Comparison

ElasticTree [5] is a state-of-the-art DCN power optimization scheme. It periodically consolidates flows based on their peak workloads with a centralized controller.

CARPO [8] is another centralized scheme for DCN power optimization, which consolidates traffic flows with low correlations together based on their 90-percentile bandwidth demands to achieve better power savings.

Optimal is the optimal solution derived by the exhaustive search approach from the same consolidation model [8] with the delay constraint. Due to its high computational complexity, *Optimal* is only evaluated in small-scale experiments.

Hier-CA Here, we also carefully design a hierarchical algorithm called Hier-CA (Hierarchical algorithm with Correlation Analysis). It decomposes the computation to different levels of DCNs.

Design. Hier-CA includes three steps. *1) Initialization.* Hier-CA begins with a priority ordering of all the M flows. *2) Core-level consolidation.* The core-level optimizer first periodically collects the data of all the M flows that pass through the core level, then conducts global correlation analysis between every pair of flows, and calculates the percentile bandwidth demands of each flow. The results are shared with the optimizer in each pod (denoted as $\{Pod_p\}$). Then, the core-level optimizer begins the consolidation by trying to set every flow to the paths following the lexicographic order. This process stops when all the M flows are set to links between the core switches and the pods. *3) Pod-level consolidation.* Each pod-level optimizer in $\{Pod_p\}$ then conducts pod-level consolidation parallelly for the M_p flows passing through itself. For a flow f_i ($1 \leqslant i \leqslant M_p$), each optimizer uses the leftmost available switches and links in the pod under the consolidation constraints. This process keeps running until all flow paths in a pod are set. Finally, all unused links and switches are put to the sleep mode to save power.

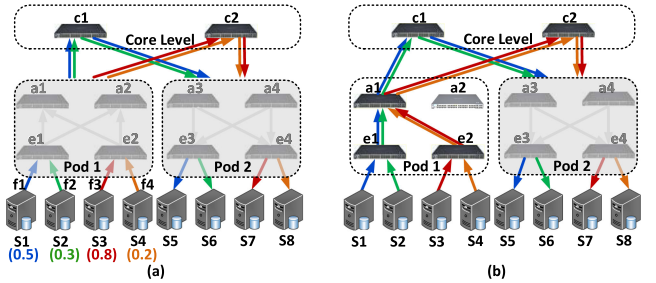


Fig. 8. Example of the Hier-CA algorithm. (a) First step: the core-level consolidation decides the flow paths only between core switches and pods. (b) Second step: the pod-level consolidation. Each pod decides the final within-pod paths for the related traffic flows.

Example. Fig. 8 shows an example with four flows (f_1 to f_4 with decreasing priority). In the initial step, the core-level optimizer collects the flows information from core switches (e.g., f_1 is from S_1 with Pod_1 to S_5 with Pod_2). It then conducts the correlation analysis for every pair of the flows, and calculates the percentile bandwidth requirement value of each flow. In the example, only f_1 and f_4 are positively correlated. These results are shared with Pod_1 and Pod_2. Then the core-level begins the consolidation (Fig. 8a) based on the consolidation constraints: f_1, f_2 are set to core switch c_1, then due to capacity limit, f_3 is set to c_2. Since f_4 and f_1 have correlation violation, f_4 is set to c_2.

Then, in the second step, Pod_1 and Pod_2 begin the pod-level consolidation in parallel to choose the specific switches and links for each flow. In Fig. 8b, under the consolidation constraints, the optimizer of Pod_1 sets all the flows to switch a_1. Thus, when all the flows are settled, the unused switch a_2 can be put to sleep to save power. At the same time, the Pod_2 optimizer independently consolidates its flows.

Limitation of Hier-CA. Even though Hier-CA tries to decompose the computation to different levels of DCNs, it is still not fully distributed and thus less scalable. In addition, it can only be applied to DCNs with a clear hierarchical topologies (e.g., Tree, Fat-tree, BCube). These limitations motivate us to design a fully distributed framework.

6.2 Experiment Setup

For the experiments, we use real DCN traffic traces from Wikipedia [26] data centers. There are 61 trace files from the 7-day Wikipedia DCN traces, each of which has a data granularity of one sample per second. We have also tested DISCO with Yahoo! DCN traces [27]. The results are skipped here due to space limitations but can be found in the conference version of this paper [37]

Testbed Setup. We set up the hardware testbed with one 48-port Open-Flow-enabled Pica8 3,290 switch (shown in Fig. 9), and six servers. To build a standard 2-pod fat-tree network topology, we configure the switch into 10 four-port virtual switches. The Open-Flow switch is connected to an independent control server. To test the baseline CARPO, we follow the setup in [8] to implement its centralized optimizer on the control server to conduct correlation-aware traffic consolidation. For DISCO, both the two algorithms have multiple sub-problem optimizers that can be deployed in a distributed way on selected switches or servers and run

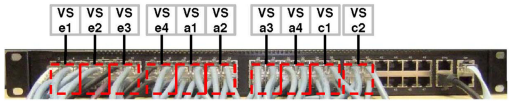


Fig. 9. Hardware testbed with 10 virtual switches (VSs) configured with a Pica8 48-port Open-Flow switch. The VSs are numbered in the same way as in Fig. 8.

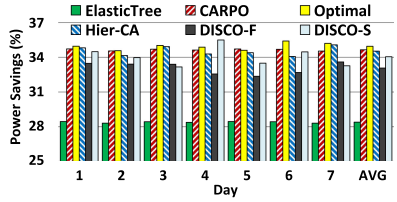


Fig. 10. Average power savings in the testbed experiments with Wikipedia traces.

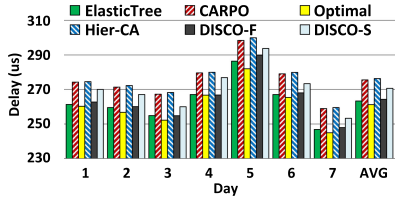


Fig. 11. Average packet delay in the testbed experiments with Wikipedia traces.

simultaneously in the DCN implementation. Since we have only one physical Open-Flow switch in our evaluation, we have to simplify the implementation to run all the sub-problem optimizers on one control server, but still in parallel. Note that DISCO can be easily implemented and extended to multiple distributed Open-Flow controllers at larger DCN scales. As the DISCO implementation discussed in Section 4, there are already multiple systems [33], [34] developed based on OpenFlow in the distributed manner, which shows the feasibility. To measure the switch power, we use the WattsUp power meter, which has the accuracy of 0.1 W with the rate of one sample per second.

In the hardware experiments, we use the 7-day Wikipedia traces as the network workloads. We randomly choose three traffic flows from the 61 Wikipedia trace files, and assign them to three pairs of servers. For a fair comparison, in all experiments, we use the same 10-minute operation period and a correlation threshold of 0.3 as used by CARPO [8]. The delay constraints are usually related to specific service requirements and DCN condition. According to the test measurement, we set RTT=100 μs and delay requirement as D_{req} = 30 ms for the flow with a length of 100 packets (on average 300 μs/packet), which is similar to [29], [31].

Simulation Setup. To investigate the performance of DISCO in large-scale DCNs, we conduct simulations with a packet-level simulator OPNET 16.1. Due to the significant amount of simulation time when the problem size increases, we simulate a 16-pod fat-tree topology (with 1,024 servers and 320 switches). For the Wikipedia flows, we duplicate 8 sets of the 61 traces and randomly choose another 24 traces (512 = 8*61+24), then randomly assigned to the 512 pairs of servers. In addition, we further conduct the simulation on a BCube (n = 8, k = 2) DCN topology with 512 servers and 192 switches. Similarly, we duplicate 4 sets of the 61 Wikipedia flow traces and randomly choose another 12 traces to form the 256 network flow traces (256 = 4*61 + 12), and randomly assigned to the 256 pairs of servers.

6.3 Hardware Testbed Results

Power Savings. The hardware results of different algorithms are shown in Fig. 10. It presents the average power saving results using the 7-day Wikipedia DCN traces as the network workloads, for a long-term evaluation. We calculate the power savings of each schemes by comparing with the original case without any traffic consolidation and power optimization. As the results shown, the average power savings of CARPO (34.6 percent) and Hier-CA (34.5 percent) are nearly the same to the Optimal baseline in each test, and outperform the other algorithms. This is due to the fact that the centralized design in CARPO and the top-level of Hier-CA can collect the information of all flows to conduct global optimization. Meanwhile, their average power savings are also better than those of the centralized ElasticTree (28.4 percent), which directly demonstrates the advantage of correlation-aware consolidation. However, note that all these centralized algorithms have scalability limitation, and can not be practical when the DCN size scales up. In addition, all of them do not consider delay.

Due to the more aggressive consolidation strategy, as well as its more limited local information, DISCO-S usually achieves more average power savings (34.0 percent here) by using fewer numbers of active switches and links, compared with that of DISCO-F (33.1 percent), but at the cost of a slightly longer average network delay (shown in Delay Performance below). Note that, power savings of both DISCO algorithms are are only 0.9 to 2.1 percent less than Optimal (34.9 percent), which are relatively small compared to the total saving percentage around 30 percent. More importantly, both DISCO-F and DISCO-S have much lower computation overhead, which are more practical for the large-scale DCNs. Besides, from the long-term evaluation (i.e., 7-day traces), the performance results of DISCO algorithms are nearly the same as those of the centralized algorithms with small variations. This demonstrates the relatively stable performance of DISCO.

Delay Performance. Accordingly, we compare the network performance in Fig. 11. It shows the average flow packet delays. *Optimal* has the shortest delay (252.2μs) that with a brutal-force path search. However, it is not feasible for large-scale DCNs due to its high computation cost. The correlation-unaware ElasticTree (263.3μs) is the second best, since it uses the peak flow demands and thus has less congestion violation. However, it also has the trade-off of much reduced power savings. Due to more aggressive consolidation, DISCO-S (270.6μs) has slightly longer average delay than DISCO-F (264.3μs). As both are designed with the delay constraints, their performances are even better than the centralized algorithms CARPO (275.6μs) and Hier-CA (276.3μs).

Note that CARPO and Hier-CA have average delays exceeding the 300 μs requirement, but both DISCO methods have better performance due to the delay constraints during the consolidation.

The testbed results show that, DISCO-F and DISCO-S can achieve slightly less or nearly the same power savings, but improved network delays performance than the centralized CARPO and the hierarchical method Hier-CA. Most importantly, DISCO algorithms are much more scalable as shown in Section 5.1.

6.4 Simulation Results on Fat-Tree Topology

To better understand the differences between DISCO and other algorithms, Fig. 12 plots their power usage traces (first 180 s in one 10 min period) during a typical run in the

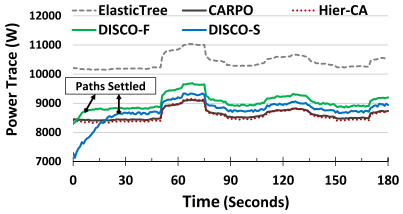


Fig. 12. Power usage traces (first 180s of one 10min period) in the simulation. DISCO algorithms adjust flow paths dynamically at the beginning, and can settle paths within a short time.

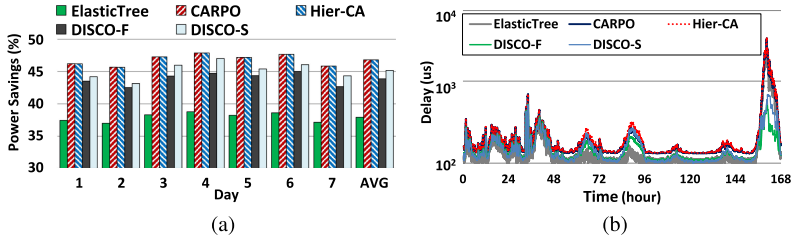


Fig. 13. Simulation with the 1024-server topology: (a) Average power savings. (b) Average packet delay variation. DISCO-S aggressively saves more power but has the trade-off of longer delay.

simulation. For ElasticTree, CARPO and Hier-CA, their flow paths are calculated with fixed numbers of active switches before the start of each period. Therefore, the change of their power traces are mainly due to the dynamic flow workloads. Different from them, at the beginning of each period, DISCO-F and DISCO-S use fewer switches with less power, under aggressive consolidation strategy. Then, when congestions are detected, DISCO algorithms adjust their flow paths dynamically by utilizing new links and switches. This is demonstrated clearly by their increasing power usages. After the settlement of paths, from Fig. 12, we can see DISCO algorithms usually have slightly higher power consumption with more active devices than the baselines. On one hand, this is due to the limitation of distributed consolidation with only local information, thus cannot achieve the near-optimal consolidation result as in the centralized algorithms. Meanwhile, DISCO also incorporates the network delay requirements, which results in more resource usage. The detailed comparison of power savings and network delay are shown as follows.

Power Savings. Fig. 13a shows the power-saving results on the 16-pod fat-tree simulation with 1,024 servers and 320 switches. From the result we can see, similar to the trend in the small-scale testbed evaluation, both CARPO and Hier-CA use only 144.6 switches on average and have the best average power savings as much as 46.8 percent, due to the centralized design with the global flow information for better optimization result. With the advantage of correlation-aware consolidation, they also have 8.9 percent more power savings than ElasticTree. Due to the limitation of local information during the path optimization as well as the delay constraints during consolidation, DISCO-S uses a larger number of switches on average (146.8) and DISCO-F uses even more (152.1). Despite their using more switches, both the two DISCO algorithms have only 1.6-2.9 percent less average power savings than CARPO and Hier-CA. Compared with the total average power savings around 43 percent, this reduction is small.

Delay Performance. The simulation results in Fig. 13b show the average package delay of the algorithms in the 7-day simulation. During the experiment, ElasticTree has the shortest delay by using peak flow workload values in the consolidation. Then it is followed by DISCO-F and DISCO-S with the delay constraints. In contrast, CARPO and Hier-CA have much longer delays, especially under heavy traffic (e.g., at 60, 80, 135 and 150 hours). Note that, even all methods have long delays on the last day, delays of both DISCO algorithms are shorter, and recover sooner than other methods.

More importantly, since the centralized schemes ElasticTree and CARPO are not feasible for the large-scale DCNs due their high computation cost, the above results demonstrate the DISCO algorithms not only are more practical for large-scale DCNs with better scalability, but also have similar or even better delay performance.

6.5 Simulation Results on BCube Topology

In this section, we further evaluate the performance of the DISCO algorithms, and present their simulation results on the BCube DCN topology. We test the performance of both DISCO-F and DISCO-S and the baselines on a BCube topology with $(n = 8, k = 2)$, which has $n^{k+1} = 512$ servers and $(k + 1)n^k = 192$ switches.

Power Savings. Fig. 15a shows the average power-saving results on the 512-server BCube topology. Compared to the result in the fat-tree topology, all the approaches have reduced power savings. This is due to the fact that BCube is a server-centric topology, which has fewer switch devices than the switch-centric fat-tree topology at a similar scale. To be specific, during the test on average CARPO and Hier-CA use 136.7 and 138.0 switches, respectively. Even though they have the top average power savings (28.8 and 28.1 percent) with the global centralized information, distributed algorithm DISCO-F and DISCO-S use 140.0 and 142.3 switches on average, and provide nearly the same average

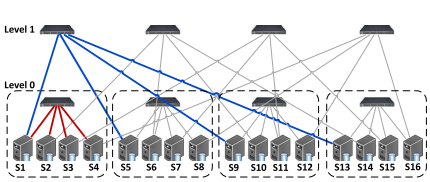


Fig. 14. Example of a Level 1 $BCube_1$ topology (k = 1, n = 4), which is formed with 4 Level 0 $BCube_0$ blocks and 4 4-port switches. The connection links on the first switch at Level 0 and Level 1 are highlighted in the colors of red and blue, respectively.

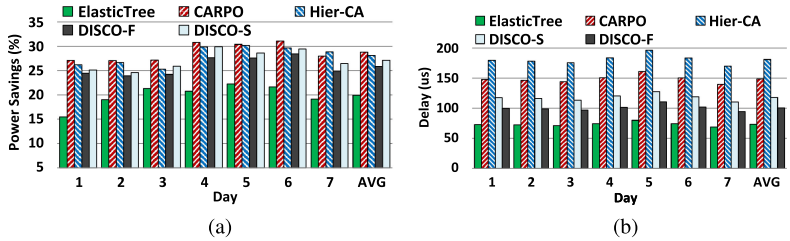


Fig. 15. Simulation with the 512-server BCube topology: (a) Average power savings. (b) Average packet delay variation. DISCO-S aggressively saves more power but has the trade-off of longer delay.

TABLE 3
Communication Overhead Comparison

	CARPO / ElasticTree	Hier-CA	DISCO-F	DISCO-S
General Case	$O(Mk^2)$	$O(Mk)$	$O(M+Q)$	$O(M)$
Simple Case	$O(k^5)$	$O(k^4)$	$O(k^3)$	$O(k^3)$

power savings (27.1 and 25.9 percent, respectively), which is only 1.0 to 2.9 percent less than CARPO and Hier-CA. ElasticTree has the least power savings as 19.9 percent without taking the advantage of correlation analysis during the traffic consolidation.

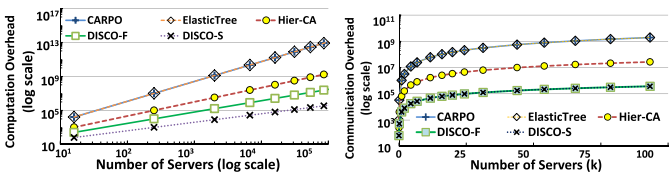
Delay Performance. Fig. 15b shows the average traffic delays with the Wikipedia traces in the BCube topology. The correlation-unaware ElasticTree (73.3μs) using peak flow demands has the shortest delay on average. DISCO-F (100.53μs) and DISCO-S (117.63μs) with the delay constraints are slightly behind, which is better than the centralized algorithms CARPO (148.47μs) and Hier-CA (180.91μs). The results demonstrate that the DISCO algorithms have shorter average delay with the flow completion time constraints. DISCO-S aggressively saves more power but it has the trade-off by having a slightly longer delay.

6.6 Communication Overhead

In order to manage the network, each DISCO optimizer (on host server or switch) needs to communicate with involved switches, which generates non-negligible communication overhead in a large-scale DCN.

Consider the example k-pod fat-tree topology. CARPO or ElasticTree needs to manage all M flows and Q switches, thus both having communication overhead of $M * Q{=}O(Mk^2)$. For Hier-CA, the overhead depends on the flows and switches number at the core and pod level, plus the flow information shared between them, which is $O(Mk)$. For DISCO-F, when the flow optimizer sets the path for a flow, its communication overhead depends on the number of switches on a flow path, which is upper bounded by Q. Moreover, when congestion occurs, the optimizer will be notified about the related flow condition for path adjustment, which is upper bounded by $O(M)$. Thus, its total overhead is $O(M{+}Q)$. For DISCO-S, each switch optimizer only collects local information, which depends on the number of flows passing through itself. Hence, its overhead is $O(M)$.

The communication overhead of all algorithms are summarized in Table 3. Fig. 16b also shows the overhead at different DCN scales. Note that, in DCNs where servers are connected with multiple flows, the above analysis is still applicable, and can be extended by adjusting the flow number M. Generally, both variants of DISCO optimizers have significantly lower overhead than CARPO/ElasticTree.



(a) Solution search space (b) Communication overhead

Fig. 16. Comparison of (a) Size of solution search space and (b) Communication overhead (both in *logarithmic scale* on Y axis) of each optimizer in all algorithms at different data center scale.

7 Conclusion

In this paper, we have presented DISCO, a highly scalable power optimization framework for large-scale DCNs, which does not depend on specific DCN architectures. DISCO features two scalable traffic consolidation algorithms that provide trade-offs between scalability, power savings, and network delay performance. First, the flow-based DISCO-F conducts consolidation on each individual flow optimizer, which provides much improved scalability but slightly less power savings. It can be used for large-scale DCNs with short flow paths or with relatively stringent delay requirements. Second, an even more scalable switch-based DISCO-S consolidates flows on each individual switch, which provides more power savings at the cost of longer network delay. It is more suitable for large-scale DCNs, where scalability is more important and a certain degree of network delay can be tolerated. We have evaluated both DISCO algorithms on a hardware testbed as well as in large-scale simulations with real DCN traces from Wikipedia and Yahoo! data centers. The results show that DISCO significantly reduces the solution search space by more than three orders of magnitude, while achieving nearly the same power savings and improved network delays compared to the state-of-the-art centralized solutions.

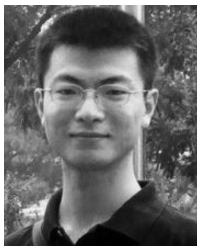
Acknowledgments

This work was supported, in part, by US National Science Foundation under Grants CNS-1421452, CCF-1758736, CNS-1758757, ECCS-1818791, and ONR N00014-17-1-2417.

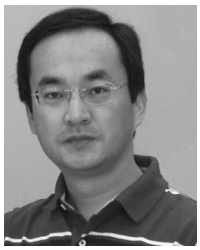
References

[1] S. Pelley *et al.*, "Understanding and abstracting total data center power," in *Proc. Workshop Energy Effcient Des.*, 2009, vol. 11, pp. 1–6.
[2] A. Greenberg *et al.*, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, 2008.
[3] A. Shehabi *et al.*, "United states data center energy usage report," Lawrence Berkeley National Laboratory Report, 2016.
[4] Data center efficiency: How we do it, 2012. [Online]. Available: http://www.google.com/about/datacenters/efficiency/internal/
[5] B. Heller *et al.*, "ElasticTree: Saving energy in data center networks," in *Proc. 7th USENIX Conf. Netw. Syst. Des. Implementation*, 2010, Art. no. 17.
[6] D. Abts *et al.*, "Energy proportional datacenter networks," in *Proc. 37th Annu. Int. Symp. Comput. Archit.*, 2010, pp. 338–347.
[7] A. Hammadi *et al.*, "Review: A survey on architectures and energy efficiency in data center networks," *Comput. Commun.*, vol. 40, pp. 1–21, 2014.
[8] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao, "CARPO: Correlation-aware power optimization in data center networks," in *Proc. IEEE INFOCOM*, 2012, pp. 1125–1133.
[9] A. Verma *et al.*, "Server workload analysis for power minimization using consolidation," in *Proc. Conf. USENIX Annu. Tech. Conf.*, 2009, Art. no. 28.
[10] A. Greenberg *et al.*, "VL2: A scalable and flexible data center network," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2009, pp. 51–62.
[11] M. Al-Fares *et al.*, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, 2008.
[12] C. Guo *et al.*, "BCube: A high performance, server-centric network architecture for modular data centers," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2009, pp. 63–74.

[13] The QFabric Architecture: Implementing a flat data center network, 2011. [Online]. Available: http://www.enpointe.com/images/pdf/The-Qfabric-Architecture.pdf
[14] J. Kim *et al.*, "Flattened butterfly: A cost-efficient topology for high-radix networks," in *Proc. 34th Annu. Int. Symp. Comput. Archit.*, 2007, pp. 126–137.
[15] N. Dukkipati *et al.*, "Why flow-completion time is the right metric for congestion control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 59–62, 2006.
[16] M. Chowdhury *et al.*, "Managing data transfers in computer clusters with orchestra," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 98–109.
[17] L. Wang *et al.*, "GreenDCN: A general framework for achieving energy efficiency in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 1, pp. 4–15, Jan. 2014.
[18] K. Zheng, X. Wang, L. Li, and X. Wang, "Joint power optimization of data center network and servers with correlation analysis," in *Proc. IEEE INFOCOM*, 2014, pp. 2598–2606.
[19] Z. Guo, S. Hui, Y. Xu, and H. J. Chao, "Dynamic flow scheduling for power-efficient data center networks," in *Proc. IEEE/ACM 24th Int. Symp. Quality Serv.*, 2016, pp. 1–10.
[20] C. Gunaratne, K. Christensen, B. Nordman, and S. Suen, "Reducing the energy consumption of Ethernet with adaptive link rate (ALR)," *IEEE Trans. Comput.*, vol. 57, no. 4, pp. 448–461, Apr. 2008.
[21] Y. Shang *et al.*, "Energy-aware routing in data center network," in *Proc. 1st ACM SIGCOMM Workshop Green Netw.*, 2010, pp. 1–8.
[22] X. Wu and X. Yang, "DARD: Distributed adaptive routing for datacenter networks," in *Proc. IEEE 32nd Int. Conf. Distrib. Comput. Syst.*, 2012, pp. 32–41.
[23] W. Cui and C. Qian, "DiFS: Distributed flow scheduling for adaptive routing in hierarchical data center networks," in *Proc. ACM/IEEE Symp. Archit. Netw. Commun. Syst.*, 2014, pp. 53–64.
[24] N. Vasić and D. Kostić, "Energy-aware traffic engineering," in *Proc. 1st Int. Conf. Energy-Efficient Comput. Netw.*, 2010, pp. 169–178.
[25] N. Vasić, P. Bhurat, D. Novaković, M. Canini, S. Shekhar, and D. Kostić, "Identifying and using energy-critical paths," in *Proc. 7th Conf. Emerg. Netw. Experiments Technol.*, 2011, pp. 1–12.
[26] G. Urdaneta *et al.*, "Wikipedia workload analysis for decentralized hosting," *Elsevier Comput. Netw.*, vol. 53, pp. 1830–1845, 2009.
[27] Y. Chen, S. Jain, V. K. Adhikari, Z. Zhang, and K. Xu, "A first look at inter-data center traffic characteristics via Yahoo! Datasets," in *Proc. IEEE INFOCOM*, 2011, pp. 1620–1628.
[28] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proc. IEEE INFOCOM*, 2011, pp. 71–75.
[29] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, pp. 63–74, 2010.
[30] M. Alizadeh *et al.*, "Analysis of DCTCP: Stability, convergence, and fairness," in *Proc. ACM SIGMETRICS Joint Int. Conf. Meas. Model. Comput. Syst.*, 2011, pp. 73–84.
[31] K. Zheng, X. Wang, and X. Wang, "PowerFCT: Power optimization of data center network with flow completion time constraints," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2015, pp. 334–343.
[32] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.
[33] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw.*, 2010, Art. no. 3.
[34] M. Yu *et al.*, "Scalable flow-based networking with DIFANE," in *Proc. ACM SIGCOMM Conf.*, 2010, pp. 351–362.
[35] J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," in *Proc. 6th Int. Conf. Syst. Syst. Eng.*, 2011, pp. 276–281.
[36] M. Amiri *et al.*, "A sequential pattern mining model for application workload prediction in cloud environment," *J. Netw. Comput. Appl.*, vol. 105, pp. 21–62, 2018.
[37] K. Zheng, X. Wang, and J. Liu, "DISCO: Distributed traffic flow consolidation for power efficient data center network," in *Proc. IFIP Netw. Conf.*, 2017, pp. 1–9.



Kuangyu Zheng received the PhD degree in computer engineering from the Ohio State University, Columbus, Ohio, in 2018, supervised by Dr. Xiaorui Wang. He is currently a faculty member of Beihang University, China. His current research interests include energy-efficient communication, computer, and network systems (e.g., data center and networks, mobile systems, IoT), Cloud\Edge\Mobile computing, 5G\6G systems and applications.



Xiaorui Wang (Senior Member, IEEE) received the DSc degree in computer science from Washington University in St. Louis, Missouri, in 2006. He is currently a professor with the Department of Electrical and Computer Engineering, Ohio State University. He is the recipient of the US Office of Naval Research (ONR) Young Investigator (YIP) Award in 2011 and the US NSF CAREER Award in 2009. He also received the Best Paper Award from the 29th IEEE Real-Time Systems Symposium (RTSS) in 2008. He is an author or co-author of more than 100 refereed publications. From 2006 to 2011, he was an assistant professor with the University of Tennessee, Knoxville, where he received the Chancellor's Award for Professional Promise and the College of Engineering Research Fellow Award. His research interests include computer architecture and systems, data center power management, and cyber-physical systems. He is the general chair of the 14th IEEE International Conference on Autonomic Computing (ICAC 2017) and the TPC co-chair of the 24th IEEE/ACM International Symposium on Quality of Service (IWQoS 2016). He is also an associate editor of the *IEEE Transactions on Parallel and Distributed Systems* (TPDS), *IEEE Transactions on Computers* (TC), and *IEEE Transactions on Cloud Computing* (TCC). He is a senior member of the IEEE Computer and Communication Societies.

Jia Liu (Senior Member, IEEE) received the PhD degree from the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, Virginia, in 2010. Since August 2017, he is an assistant professor with the Department of Computer Science, Iowa State University. He was a postdoctoral researcher from February 2010 to November 2014, and subsequently a research assistant professor from November 2014 to July 2017, both with the Department of Electrical and Computer Engineering, Ohio State University. His research areas include theoretical foundations of control and optimization for stochastic networked systems, distributed algorithms design, optimization of cyber-physical systems, Internet-of-Things, data analytics infrastructure, and machine learning. He is a member of the ACM. His work has received numerous awards at top venues, including IEEE INFOCOM'16 Best Paper Award, IEEE INFOCOM'13 Best Paper Runner-up Award, IEEE INFOCOM'11 Best Paper Runner-up Award, and IEEE ICC'08 Best Paper Award. He is a recipient of Bell Labs President Gold Award in 2001 and China National Award for Outstanding PhD Students Abroad in 2008. His research has been supported by NSF, AFOSR, AFRL, and ONR.