DISCO: Distributed Traffic Flow Consolidation for Power Efficient Data Center Network

Kuangyu Zheng, Xiaorui Wang, and Jia Liu
The Ohio State University, Columbus, OH, USA

***Abstract*—Power optimization for data center networks (DCNs) has recently received increasing research attention, since a DCN can account for up to 20% 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 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 paper, we propose DISCO, a DIStributed traffic flow COnsolidation framework, with correlation analysis and delay constraints, for large-scale power efficient 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 with more than three orders of magnitude smaller problem size for individual optimizers (10^4 to 10^6 times faster for a DCN at the scale of 10K servers). The convergence of DISCO is also proved theoretically and evaluated experimentally.

I. 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]. As the power efficiency of data center cooling has been significantly improved in recent years (e.g., by using cold river water for cooling [3]), 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 up to 20% of the total power consumption of a data center [2][4], has started to receive increasing attention [4][5].

Among the recent studies of DCN power optimization strategies, one of the most effective approaches is based on the idea termed *dynamic traffic consolidation* [4][6], 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 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 [7], 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) [4][6].

While traffic consolidation has shown great promise, existing solutions are mostly *centralized* and do not scale well for large-scale DCNs. For example, ElasticTree [4] employs a centralized optimization framework, where all links and switches in the DCN are considered by a single centralized optimizer for consolidation. Likewise, CARPO [6] 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 I, it could take more than three to five hours to finish one round of consolidation for a normal-size DCN with 10K servers. Hence, the performance of these existing schemes become unacceptable in DCNs with hundreds of thousands of servers. As a result, highly scalable traffic consolidation algorithms are much needed for future DCNs whose sizes are expected to grow rapidly [8][5].

TABLE I: Computation time[1]at different DCN scales

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

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 savings can be close to that of a centralized scheme, while the desired scalability could be achieved. Second, there are various types of DCN architectures [5] that may require different decomposition schemes. For example, DCNs can be categorized as hierarchical ones (e.g., fat-tree [9], VL2 [8], and BCube [10]) and non-hierarchical ones (e.g., QFabric [11] and FBFLY [12]). In fat-tree (e.g., Figure 4), a typical hierarchical DCN, switches are organized at three levels: core, aggregation, and edge. This leads to the straightforward idea to decompose

This work was supported, in part, by NSF under grant CNS-1421452.

[1]Based on our lab server (CPU: Intel Dual Core 3.3GHz; RAM: 4GB).

the global problem according to switch levels. However, this strategy is not applicable for other DCN architectures without such a well-established hierarchy. Therefore, how to design efficient decomposition schemes that are suitable for different DCN architectures is non-trivial and remains an open problem.

In this paper, we propose DISCO, a DIStributed power optimization framework based on traffic COnsolidation for large-scale DCNs. Similar to existing works [7][6], 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 tradeoffs (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. Second, an even more scalable switch-based algorithm (DISCO-S) is proposed to consolidate flows on each individual switch in a distributed fashion with more aggressive power savings. In addition, since the network delay performance has been identified as an important metric in current DCN services [13], delay constraints are included in the traffic consolidation to enforce the network performance of DISCO. 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 more than three orders of magnitude smaller problem size for individual optimizers. We also discuss the trade-offs and applicable DCN scenarios of each algorithm.
- We enforce network delay constraints into the traffic consolidation of DISCO. It significantly improves the DCN delay performance, compared to previous schemes that ignored this important metric.
- We evaluate DISCO both on a hardware testbed and in large-scale simulations with real DCN traces. Our results show that DISCO can have nearly the same power savings and network delay performance, compared with the centralized solution.
- We theoretically prove that the distributed designs of DISCO converge to a stable state in polynomial time. Experimental results also demonstrate that DISCO has a short convergence time.

In the rest of the paper: Section II discusses the related work. Section III provides the background of correlation analysis. Section IV introduces the design of the proposed DISCO algorithms. Section V presents the baselines for comparison. Section VI analyzes the convergence of DISCO. Section VII presents the evaluation with the experiment results. Section VIII concludes the paper.

II. Related Work

There are several existing approaches designed for DCN power optimization [4][6][14]. One approach is link adaptation [15], 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 [4][6][16]. We note, however, that these existing works are centralized schemes, which require global information and do not scale well (i.e., high computational complexity and large 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 [17][18][19]. For example, DARD [17] and DiFS [18] propose distributed adaptive routing methods to achieve load balancing by moving elephant flows from overloaded paths to underloaded paths. One related DCN power optimization work is HERO [20], which uses a hierarchical scheme to reduce problem size for individual optimizer. However, HERO is still partly centralized and designed for certain hierarchical DCNs only. In contrast, DISCO algorithms are fully distributed with better scalability, and they are not dependent on any specific type of DCN topology. In addition, DISCO utilizes the correlation-aware consolidation, which has been proven to provide much better power savings [7][6].

III. Background on Correlation Analysis

We now briefly introduce the concept of correlation-aware traffic consolidation. It was first proposed by [7] and has been shown to provide over 20% more power savings [6] than traditional method like ElasticTree [4], which will serve as the foundation of our DISCO power optimization design.

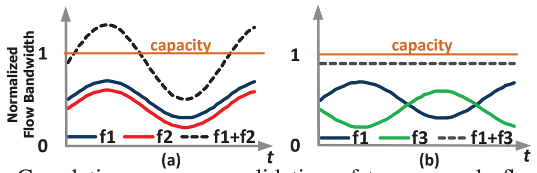


Fig. 1: Correlation-aware consolidation of two example flows leads to lower capacity requirement and so better consolidation (the dashed line is the aggregated load). (a) Positively correlated, with variation magnified in the aggregation. (b) Negatively correlated, with variation canceled in the aggregation.

Figure 1 compares two traffic consolidation examples: positively correlated flows and negatively correlated flows. The flows are normalized to the link capacity. In Figure 1(a), 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 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 Figure 1(b)), their sum will have less variation, thus requiring less capacity for consolidation. The correlation degree can be quantified by the Pearson Correlation Coefficient [6].

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 [21][22], and 2) most of the loads

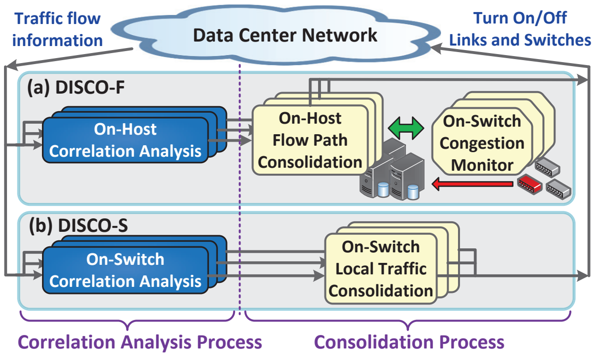

Traffic flow information
Data Center Network
Turn On/Off Links and Switches
(a) DISCO-F
On-Host Correlation Analysis
On-Host Flow Path Consolidation
On-Switch Congestion Monitor
(b) DISCO-S
On-Switch Correlation Analysis
On-Switch Local Traffic Consolidation
Correlation Analysis Process
Consolidation Process

Fig. 2: The DISCO framework includes two distributed 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.

of different traffic flows are weakly correlated, which means they do not always peak at the same time [6].

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 (to avoid positive correlation). This approach has been demonstrated to yield more power savings [6][7]. Different from previous work that focus mostly on centralized correlation-aware traffic or server consolidation, in this paper, our DISCO algorithms decompose correlation analysis into localized power optimizers for highly scalable traffic consolidation in large-scale DCNs.

IV. Design of DISCO

In this section, we first introduce the DISCO optimization framework that includes two distributed traffic consolidation algorithms, DISCO-F and DISCO-S. We then explain the two algorithms in detail with illustrative examples.

A. *The DISCO Framework*

Traditional centralized traffic consolidation algorithms usually start with the analysis of traffic information of the entire DCN, including the bandwidth requirements of the flows and the link conditions. They then periodically route the flows to share the same paths under the link capacity constraints so that only a small set of switches and links are needed, and the unused devices can be dynamically put to sleep to save power (and woken up later when the workload increases). But in a large-scale DCN, both the amount of traffic information and the consolidation computation increase dramatically with the DCN size, which makes the centralized algorithms infeasible. Therefore, the computational problem size needs to be reduced by decomposing the centralized problem into a set of smaller optimization sub-problems for scalability. Since the traffic consolidation problem is known to be NP-hard [6][23], we developed the feasible approximation algorithm DISCO with strong performance guarantee.

Before diving into details, we first provide an overview of the two proposed algorithms. As the framework illustrates in Figure 2, both DISCO algorithms share the same general processes of correlation analysis and traffic consolidation, which are performed periodically to adapt the traffic variations. Different from traditional methods, DISCO further incorporates the network delay constraints during the traffic consolidation to improve the delay performance. In addition, even the traffics have unpredictable changes, DISCO algorithms are adaptive to flow path adjustments. Between the two algorithms, the operator can choose the desired one based on the specific DCN architecture and desired trade-offs between scalability, power savings, and network delay performance.

Enforced delay constraints: DISCO incorporates the delay constraints during the traffic consolidation based on the DCTCP protocol [24]. DCTCP leverages the Explicit Congestion Notification (ECN) function by setting a clear queuing packet number notification threshold K, and can provide more fair bandwidth sharing and less queuing delay. It is proved that the steady-state queue-length is at most as Q_{MAX}=K+N_s*W_s [24], 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 [25], 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 switch number n on a candidate path, and only chooses the path satisfied the delay constraint as $FCT_j \leq D_{req}$, where D_{req} is the delay requirement.

Flow-based: 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 to 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 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 problem size of solution search space dependents on the length of the path. This problem size could still be high in a large-scale DCN that has many long-path flows.

Switch-based: To further reduce the problem size and achieve better power savings, we propose an even 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

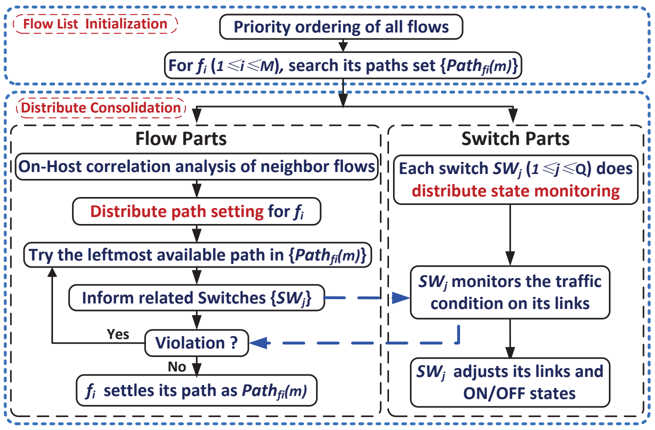

Flow List Initialization
Priority ordering of all flows
For f_i ($1 \leqslant i \leqslant M$), search its paths set {$Path_{fi}(m)$}
Distribute Consolidation
Flow Parts
Switch Parts
On-Host correlation analysis of neighbor flows
Distribute path setting for f_i
Try the leftmost available path in {$Path_{fi}(m)$}
Inform related Switches {SW_j}
Violation ?
Yes
No
f_i settles its path as $Path_{fi}(m)$
Each switch SW_j ($1 \leqslant j \leqslant Q$) does distribute state monitoring
SW_j monitors the traffic condition on its links
SW_j adjusts its links and ON/OFF states

Fig. 3: The algorithm flow chart of algorithm DISCO-F.

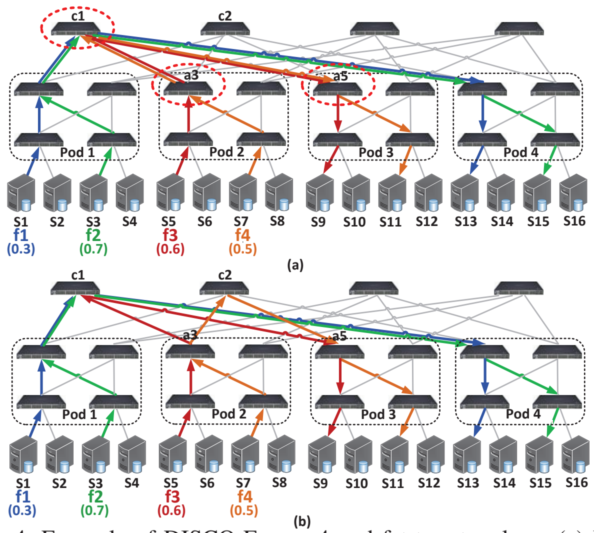


Fig. 4: Example of DISCO-F on a 4-pod fat-tree topology. (a) First step: Each flow is individually consolidated to the leftmost side of the network for power-saving in a distributed way. Congestion occurs on the dash-circled switches. (b) Second step: With the congestion notification from switches c_1, 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.

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 also does not depend on any particular type of DCN structures.

Note that, since each switch optimizer only has local information, DISCO-S may also yield sub-optimal solutions. Moreover, different distributed optimizers may have conflicts in decision making, thus globally there could be more path adjustments due to congestion. Therefore, compared with DISCO-F, the trade-offs for DISCO-S are potentially more network delay and a longer convergence time.

Generally, DISCO-F can be used for large-scale DCNs with relatively short flow paths or with more stringent delay requirements. DISCO-S is more suitable for very large-scale DCNs, where scalability is more important (e.g., when there is a significantly large number of flows that even number of DISCO-F controllers becomes a bottleneck) and a certain degree of network delays can be tolerated.

DISCO Implementation: For DISCO-F, the flow optimizer can be implemented on its source server. For DISCO-S, the optimizer can run on the processor of each switch. Moreover, current software defined network (SDN) switches (e.g., Open-Flow based) 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 most of current Open-Flow systems work in a centralized scheme, we propose to utilize Open-Flow technology distributively to address the scalability issue. Currently, some Open-Flow based systems are already developed in a distributed manner by having multiple controllers for different switch groups [26][27], 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.

B. DISCO-F: Flow-based Algorithm

DISCO-F includes two parts as shown in Figure 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. Similar to [4][6], only paths without loops are considered. 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.

Meanwhile, we assume there is a priority order of the DCN traffic flows, which can be based on different service types, importance, or bandwidth requirements [28][6]. 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 the options are tried.

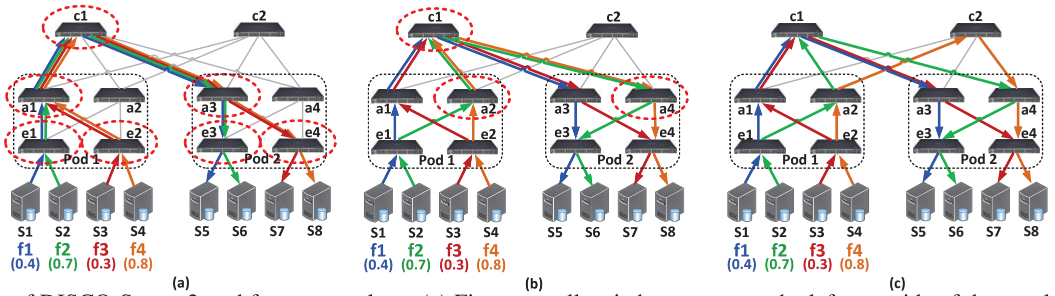


Fig. 5: Example of DISCO-S on a 2-pod fat-tree topology. (a) First step: all switches try to use the leftmost side of the topology distributively. Congestion occurs on some switches (in red dash circle). (b) Second step: congested switches do path adjustments beginning from flows with lower priorities. Paths of f_2 and f_4 are adjusted. However new congestion 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.

The convergence of this process will be proved in Section VI.

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 VI).

Example: Figure 4 shows a 4-pod fat-tree with four flows (without loss of generality, we assume that f_1 to f_4 with the 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 Figure 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 send 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 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 (Figure 4(a)) shows congestions occur on switches c_1, a_3 and a_5 (in red-dashed circle) due to the lack of knowledge of the other flow paths. Then, the congested switches check the passing 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 (Figure 4(b)), the optimizer of f_4 on S_7 updates its path to the next leftmost path, which resolves the congestion.

C. *DISCO-S: Switch-based Algorithm*

DISCO-S is also a fully distributed algorithm, in which every switch performs traffic consolidation individually. As shown in Figure 6, the switch optimizer on SW_j starts the correlation analysis only among the flows that pass itself. Since flow paths without loops can be calculated and stored in the forward table of each switch in advance, only links on these paths are considered. Usually, there are multiple available forwarding links for each flow based on its destination, SW_j performs traffic consolidation by choosing links in the lexicographic order. When all passing flows f_{ij} are settled, if SW_j is unused, it will be put into sleep until the beginning of the next period. Similar to DISCO-F, this decision is made after the global convergence time (Section VI). Note that, due to the local correlation analysis and limited path information, each switch can only choose the flow path according to its own 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 have more power savings but with relatively longer adjustment time.

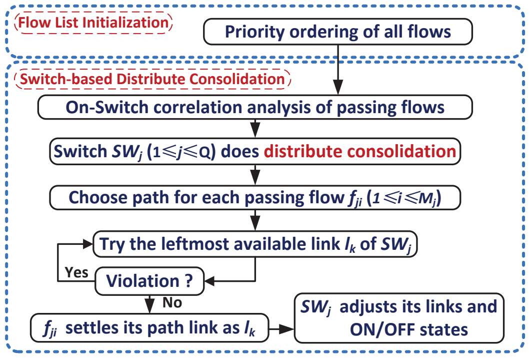

Flow List Initialization
Priority ordering of all flows
Switch-based Distribute Consolidation
On-Switch correlation analysis of passing flows
Switch SW_j ($1 \leqslant j \leqslant Q$) does distribute consolidation
Choose path for each passing flow f_{ji} ($1 \leqslant i \leqslant M_j$)
Try the leftmost available link l_k of SW_j
Yes
Violation ?
No
f_{ji} settles its path link as l_k
SW_j adjusts its links and ON/OFF states

Fig. 6: The algorithm flow chart of the algorithm DISCO-S.

Example: Figure 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 Figure 5(a). In the second step (Figure 5(b)), 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 Figure 5(b). 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 Figure 5(c).

V. Baseline Designs

In this section, we introduce the baselines that are used in this paper for performance comparison.

A. *Centralized baselines*

ElasticTree [4] is a state-of-the-art DCN power optimization scheme. We use its heuristic version which periodically

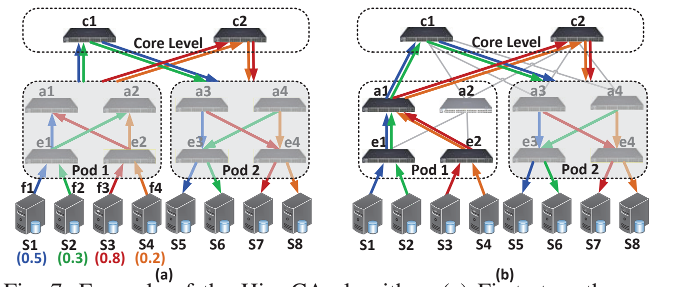


Fig. 7: Example of the Hier-CA algorithm. (a) First step: the core-level consolidation decides the flow paths only between core switches and pods. (b) Second step: the pod-level consolidation. Each pod decides the final within-pod paths for the related traffic flows.

consolidates flows based on their peak workloads with a centralized controller.

CARPO [6] 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.

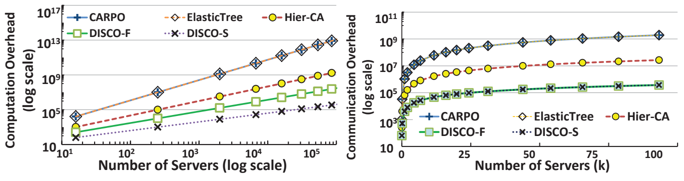
Optimal is the optimal solution derived by the exhaustive search approach from the same consolidation model [6] with the delay constraint. Due to its high computational complexity, *Optimal* is only evaluated in the small-scale experiments.

B. Hier-CA

We also designed a hierarchical baseline called Hier-CA (Hierarchical algorithm with Correlation Analysis). It decomposes the computation to different levels of DCNs. However, Hier-CA is not fully distributed and can only be applied to DCNs with a clear hierarchical topologies.

Design: Hier-CA begins with an initialization by ordering priority of the M flows. It then applies the consolidation by different levels. In the core-level, the optimizer periodically conducts global correlation analysis of all flows, and applies traffic consolidation by trying to set every flow to the path links between the core switches and the pods following the lexicographic order. After that, each optimizer in $\{Pod_p\}$ conducts pod-level consolidation in parallel for the M_p flows passing through, trying to use the leftmost available switches and links in pod under the consolidation constraints. Then, unused links and switches are put into the sleep mode to save power until the beginning of the next period.

Example: Figure 7 shows an example (f_1 to f_4 with decreasing priority). First, the core-level optimizer collects the flows information (e.g., f_1 is from S_1 with Pod_1 to S_5 with Pod_2) and conducts the correlation analysis for every flow pairs. In the example, only f_1 and f_4 are positively correlated. Then the core-level begins the consolidation (Figure 7(a)) 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. Secondly, Pod_1 and Pod_2 begin the pod-level consolidation in parallel. In Figure 7(b), under the 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.



(a) Solution search space (b) Communication overhead

Fig. 8: 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.

TABLE II: 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	$O(k^7)$	$O(k^5)$	$O(k^4)$	$O(k^3)$

VI. Theoretical Analysis

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

A. 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 [26][27]. Meanwhile, the communication overhead incurred by the optimizers (e.g., to collect and exchange information) is also an importance factor for scalability [17][18]. 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. For illustration, we first assume a simple network setup as follows: 1) there is one traffic flow between a pair of servers, and 2) each server 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 can still hold.

Centralized CARPO and ElasticTree decide 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 decides the core switches to use, with a search space of $O(Mk^2)$. The pod level optimizer decides the local flow paths with a search space of $O(M_{Pmax}k/2)$ ($M_{Pmax}\leqslant k^2$/4 in the example case). Therefore, the problem size 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 II summarizes the sizes of solution search space of different algorithms. Compared with CARPO, DISCO-F successfully reduces the solution search space by *at least three*

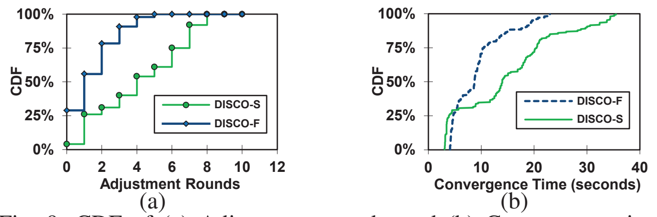


Fig. 9: CDF of (a) Adjustment rounds, and (b) Convergence time, for all the optimizers (512 in DISCO-F and 320 in DISCO-S) in 1024-server simulation.

orders of magnitude of k. As shown with the **log scale** in Figure 8(a), 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.

Communication Overhead Analysis: 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 the local information, which depends on the number of flows passing through itself. Hence, its overhead is $O(M)$.

The communication overheads of all algorithms are summarized in Table III. Figure 8(b) 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, the two variants of DISCO optimizers have significantly lower overhead than CARPO/ElasticTree.

B. Convergence Analysis

Due to the distributed designs, both DISCO-F and DISCO-S may have transient flow congestions (or delay constraint violations), and need path adjustment for multiple rounds. Hence, it is important to 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 for adopting the percentile value for the correlation analysis in Section III.

Theoretically, we also model and prove the convergence

TABLE III: 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	$O(k^5)$	$O(k^4)$	$O(k^3)$	$O(k^3)$

feature of both DISCO-F and DISCO-S for general DCN topologies. Due to space limits, only the analysis of DISCO-F is shown here. The full proof for all algorithms are presented in our technical report [29].

Analysis of DISCO-F. When SW_x detects congestion on any incident link $l(x,y)$ among all the M_x passing flows $\{f_i\}$, DISCO-F optimizer removes flows starting from the lowest priority, i.e., $i = M_x$, M_x-1, M_x-2, ..., until $l(x,y)$ is not congested or all path options have been tested.

In the initialization stage, all DISCO-F optimizers allocate each f_i to its first path $Path_{f_i}(1)$. When f_i needs to adjust its path, the optimizer moves it from $Path_{f_i}(m)$ to the next $Path_{f_i}(m{+}1)$, unless $Path_{f_i}(m)$ is already the last available path ($m{=}P_i$). Under this process, we have the following result:

Proposition 1. The congestion adjustment of DISCO-F terminates in polynomial time.

Proof: Consider the worst case with the largest possible amount of adjustments, where all the M flows have congestion, and each f_i needs to test the largest possible rounds on its available paths. Without loss of generality, we assume that flows are sorted in a decreasing order of priority.

In the initial state, each f_i uses its first path $Path_{f_i}(1) \in \{Path_{f_i}(m)\}$, $1{\leqslant}m{\leqslant}P_i$. Since only a finite number of flows with lower priorities need to be adjusted, after the first round, at least f_1 will settle its path as $Path_{f_1}(1)$, and all the other f_i ($2{\leqslant}i{\leqslant}M$) are adjusted to their next paths $Path_{f_i}(2)$. Similarly, in the second round, f_2 will be settled to $Path_{f_2}(2)$, and the rest M-2 flows are moved to their next paths. Therefore, in the ith round, only remained M-i flows need further adjustments with reduced search space of P_i-i path choices. Since the workload of each flow is considered quasi-stationary within a period, once the paths of flows with higher priorities are settled, they do not need to be changed in this period.

Thus, in the worst case, each optimizer will stop after at most M rounds and become stable. After all optimizers terminate adjustment, which is reachable as shown above, the whole system becomes stable. For DISCO-F, the global search space of the path options to be tested is upper bounded by $O(M{\cdot}\{P_i\}_{MAX})$. As a special case, for the k-pod fat-tree topology, $P_i{\leqslant}(k/2)^2$. So the total global search space of path options for DISCO-F is upper bounded by $O(Mk^2)$.

Therefore, for DISCO-F, in each round there will be fewer flows to be adjusted, and the remaining path set search space is also reduced. Thus, DISCO-F will terminate within a finite number of rounds. This completes the proof. ■

Figure 9(a) compares the Cumulative Distribution Function (CDF) of the number of rounds for convergence of each optimizer in the 1024-server simulation (setting details in Section VII-A). In DISCO-F, about 90% of the flows finish the adjustment within 3 rounds, while in DISCO-S, it needs 7 rounds to settle 90% of switches.

Convergence speed is another important performance metric for distributed algorithms. As the experiment result of DCN with 1024 servers (setting details in Section VII-A) shown in Figure 9(b), 90% of the flows in DISCO-F can converge within 20 seconds. The convergence time of DISCO-S is longer: 90% of the switches finish adjustments within about 33 seconds. 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 III) the flow conditions are stable within each period, which is consistent with previous work [4][6]. However, if flows are shorter/longer in other scenarios, the algorithm period can be adjusted according to the specific cases, while the convergence of algorithms should still hold.

VII. Experiment Evaluation

In this section, we evaluate DISCO in terms of power savings and network performance (i.e., packet transmission delay) on a hardware testbed and in large-scale simulation.

A. *Experiment Setup*

For the experiments, we use real DCN traffic traces from Wikipedia [21] and Yahoo! DCP [22] 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. The Yahoo! DCP traces are of 24 hours and have 70 traces with the same sampling granularity.

Testbed Setup: We set up the hardware testbed with one 48-port Open-Flow-enabled Pica8 3290 switch (shown in Figure 10), and six servers. To build a 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 [6] 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 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 IV, there are already multiple systems [27] [26] developed based on Open-Flow in the distributed manner, which shows the feasibility. The switch power is measured with the WattsUp power meter (accuracy: 0.1W, 1sps).

In the first set of 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 the three pairs of servers. In the second set of hardware experiments, we evaluate all the algorithms using the Yahoo! DCN traces, which on average have heavier loads than the Wikipedia traces. Since the Yahoo! traces only last for 24 hours, we randomly choose three sets, each with three traffic flows from the 70 Yahoo! traces, and assign them to the three pairs of servers. For a fair comparison, in all experiments, we use the same 10-minute operation period and a

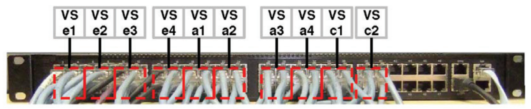


Fig. 10: Hardware testbed with 10 virtual switches (VSs) configured with a production Pica8 48-port Open-Flow switch. The VSs are numbered in the same way as in Figure 7.

correlation threshold of 0.3 as used by CARPO [6]. The delay constraints are usually related to specific service requirements and DCN condition. According to the test measurement, we set RTT=100 μs and the 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 [24][25]. When applying DISCO to other DCN scenarios, these parameters (e.g., period, threshold) can be adjusted according to specific requirements.

Simulation Setup: To investigate the performance of DISCO in large-scale DCNs, we conduct the simulations with a packet-level simulator OPNET 16.1. Due to the significant amount of simulation time when the problem size increases, we only simulate a 16-pod fat-tree topology (with 1024 servers and 320 switches). For the Wikipedia flows, we duplicate 8 sets of the 61 traces and randomly choose another 24 traces, then randomly assigned to the 512 pairs of servers.

B. *Hardware Testbed Results*

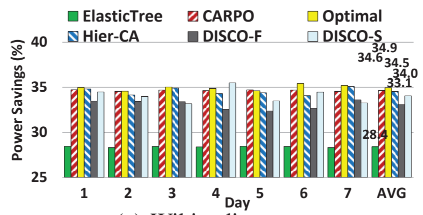
Power Savings: The hardware results are shown in Figure 11 (a) and (b), respectively. In the two experiments, the power savings of CARPO (34.6% and 34.2%) and Hier-CA (34.5% and 34.2%) are nearly the same, which are better than those of ElasticTree (28.4% and 28.8%) due to correlation-aware consolidation. However, all of them do not have delay consideration. The power savings of DISCO-F (33.1% and 31.6%) and DISCO-S (34.0% and 32.7%) enforce the delay constraints with the tradeoff as less power savings. However, they are only 0.9% to 2.1% less than *Optimal* (34.9% and 34.6%), but with much lower computation overhead.

Delay Performance: Figure 12(a) shows the average traffic delays with the Wikipedia traces. *Optimal* has the shortest delay (252.2μs). The correlation-unaware ElasticTree (263.3μs) using peak flow demands is the second best. Then DISCO-F (264.3μs) and DISCO-S (270.6μs) with the delay constraints are closely behind, which is even better than the centralized algorithms CARPO (275.6μs) and Hier-CA (276.3μs). Similar results appear in Figure 12(b) with Yahoo! traces. Note that CARPO and Hier-CA have average delays over the 300 μs requirement, but both DISCO methods have better chances to meet the delay constraints.

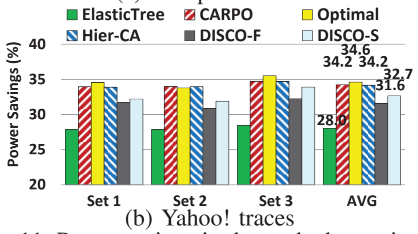
These results show that, in the small-scale testbed experiments, DISCO-F and DISCO-S can achieve slightly less or nearly the same power savings, but improved delays performance than the centralized CARPO and the hierarchical method Hier-CA, while most importantly being much more scalable as shown in Section VI-A.

C. *Simulation Results*

Power Savings: Figure 13(a) shows the power-saving results on the 16-pod fat-tree simulation with 1024 servers and 320 switches. Both CARPO and Hier-CA use 144.6 switches on average, with the power savings as much as 46.8%, which is 8.9% more than that of ElasticTree. Due to the lack of

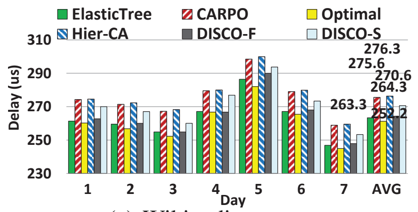


(a) Wikipedia traces

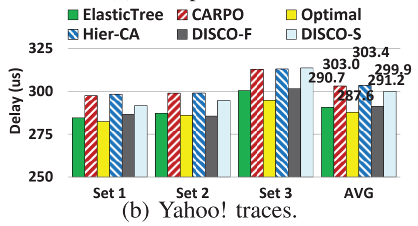


(b) Yahoo! traces

Fig. 11: Power savings in the testbed experiments with traces of Wikipedia (a) and Yahoo! (b). Hier-CA and DISCO-F are close to CARPO; DISCO-S aggressively saves more power, but results in longer delay in Fig.12 (a) and (b).

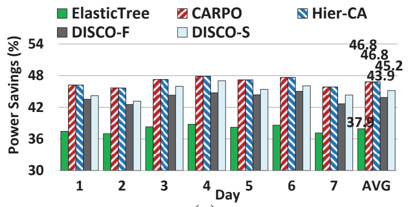


(a) Wikipedia traces.

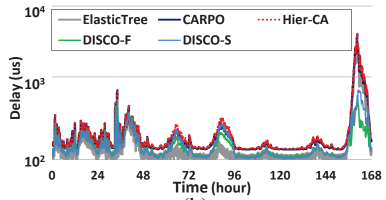


(b) Yahoo! traces.

Fig. 12: Average packet delay in the testbed experiments with Wikipedia traces (a), and Yahoo! traces (b).



(a)



(b)

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

global information in the path optimization as well as the delay constraints, DISCO-F and DISCO-S use more switches on average (152.1 and 146.8, respectively), but with only 1.6-2.9% less power savings than CARPO and Hier-CA.

Delay Performance: The simulation results in Figure 13(b) show the average package delays of different algorithms in the 7-day simulation. During the whole experiment, ElasticTree has the shortest delay by using peak 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 delay, especially under heavy traffic (e.g., at 60, 80, 135 and 150 hours). Note that even all methods have long delay on the last day, delays of both DISCO algorithms are shorter, and recover sooner than other methods.

VIII. 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 distributed traffic consolidation algorithms, i.e., flow-based DISCO-F and switch-based DISCO-S, that provide trade-offs between scalability, power savings, and network delay performance. DISCO algorithms are evaluated on a hardware testbed as well as in large-scale simulations with real DCN traces from Wikipedia and Yahoo! DCNs. 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.

References

[1] S. Pelley, D. Meisner, T. F. Wenisch, and J. VanGilder, "Understanding and abstracting total data center power," in *WEED*, 2009.

[2] A. Greenberg *et al.*, "The cost of a cloud: Research problems in data center networks," *SIGCOMM CCR.*, vol. 39, no. 1, Dec. 2008.

[3] "Data center efficiency: How we do it," 2012. [Online]. Available: http://www.google.com/about/datacenters/efficiency/internal/

[4] B. Heller *et al.*, "ElasticTree: Saving energy in data center networks," in *NSDI*, 2010.

[5] A. Hammadi *et al.*, "Review: A survey on architectures and energy efficiency in data center networks," *Comput. Commun.*, 2014.

[6] X. Wang *et al.*, "Correlation-aware traffic consolidation for power optimization of data center networks," *IEEE Transactions on Parallel and Distributed Systems*, 2016.

[7] A. Verma *et al.*, "Server workload analysis for power minimization using consolidation," in *USENIX*, 2009.

[8] A. Greenberg *et al.*, "VL2: a scalable and flexible data center network," in *SIGCOMM*, 2009.

[9] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *SIGCOMM*, 2008.

[10] C. Guo *et al.*, "BCube: a high performance, server-centric network architecture for modular data centers," in *SIGCOMM*, 2009.

[11] "The QFabric Architecture: Implementing a flat data center network." [Online]. Available: http://www.enpointe.com/images/pdf/The-Qfabric-Architecture.pdf

[12] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: A cost-efficient topology for high-radix networks," in *ISCA*, 2007.

[13] M. Chowdhury *et al.*, "Managing data transfers in computer clusters with orchestra," in *SIGCOMM*, 2011.

[14] K. Zheng and X. Wang, "Dynamic control of flow completion time for power efficiency of data center networks," in *ICDCS*, 2017.

[15] C. Gunaratne *et al.*, "Reducing the energy consumption of Ethernet with adaptive link rate (ALR)," *Computers, IEEE Transactions on*, 2008.

[16] Y. Shang, D. Li, and M. Xu, "Energy-aware routing in data center network," in *SIGCOMM Workshop on Green Networking*, 2010.

[17] X. Wu and X. Yang, "DARD: Distributed adaptive routing for datacenter networks." in *ICDCS*, 2012.

[18] W. Cui and C. Qian, "DiFS: Distributed flow scheduling for adaptive routing in hierarchical data center networks," in *ANCS*, 2014.

[19] B. Genge *et al.*, "A hierarchical control plane for software-defined networks-based industrial control systems," in *IFIP Networking*, 2016.

[20] Y. Zhang and N. Ansari, "HERO: Hierarchical energy optimization for data center networks." in *ICC*, 2012.

[21] G. Urdaneta, G. Pierre, and M. van Steen, "Wikipedia workload analysis for decentralized hosting," *Elsevier Computer Networks*, 2009.

[22] Y. Chen *et al.*, "A first look at inter-data center traffic characteristics via yahoo! datasets," in *INFOCOM*, 2011.

[23] M. Wang, X. Meng, and Z. Li, "Consolidating virtual machines with dynamic bandwidth demand in data centers." in *INFOCOM*, 2011.

[24] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," in *SIGCOMM*, 2010.

[25] K. Zheng *et al.*, "PowerFCT: Power optimization of data centernetwork with flow completion time constraints," in *IPDPS*, 2015.

[26] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," in *SIGCOMM*, 2010.

[27] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *INM/WREN'10*, 2010.

[28] M. Alizadeh *et al.*, "pFabric: Minimal near-optimal datacenter transport," in *SIGCOMM*, 2013.

[29] "DISCO technical report," 2017. [Online]. Available: https://www.dropbox.com/s/9i9c2kidgz7pwem/DISCO-TechReport.pdf