Consolidating Flows with Implicit Deadlines for Energy-Proportional Data Center Networks

Xiaoda Zhang*, Zhuzhong Qian*, Sheng Zhang*, Kui Wu†, Sanglu Lu*
*State Key Laboratory for Novel Software Technology, Nanjing University, China
†Department of Computer Science, University of Victoria, Canada
Email: zhangxiaoda@dislab.nju.edu.cn, {qzz, sheng, sanglu}@nju.edu.cn, wkui@cs.uvic.ca

***Abstract*—To reduce the energy consumption of a large number of network devices in a data center, energy-efficient schemes use various heuristics to consolidate traffic to fewer switches. Most of these works, however, ignore the flow-level performance, which is one of the most critical requirements in production data centers. Hence, flow rate allocation should be considered together with flow path selection to guarantee flow-level performance and in the meantime save energy of network devices. For this reason, we present a framework to ensure that the energy consumption for data center network (DCN) is proportional to the traffic and to guarantee the flow-level performance. Our solution consists of two components: (i) flow rate allocation to meet flows' deadlines and (ii) flow path selection to use fewer switches. We compare our framework with existing techniques under synthetic traffic patterns. Results show that our framework could save, on average, 20% of network energy than the always-on baseline, while maintaining the better flow-level performance, and achieving good running time and fault tolerance simultaneously.**

I. Introduction

High energy consumption has become a central issue for large-scale data centers as computing and networking infrastructures scale out in response to growing requests in clouds. It is shown in [1] that the energy used in U.S. data centers in 2013 was estimated 91 billion kwh and is projected to increase to roughly 140 billion kwh annually by 2020. Various techniques, including Dynamic Voltage Frequency Scaling (DVFS), virtualization and efficient power supplies, have been explored to reduce the energy consumed by servers which accounts for the largest component of a data center's total power. The network devices, which could account for 10-20% of a data center's total power [9], [13], also need energy saving schemes.

DCNs are usually provisioned for the peak workload, and this load exceeds their long-term utilizations by a large margin. Therefore, significant energy could be saved if the energy consumption of a DCN could be proportional to its actual rather than peak workload. We call a DCN with this feature an *energy propoertional* DCN. A power measurement [16] studied several data center switches under a variety of traffic patterns. The study showed that keeping a switch always on consumes most energy, while increasing the traffic from zero to full load via a switch only increases the switch's power by less than 10%. This phenomenon implies that to achieve DCN energy proportionality, we could mainly focus on the number of power-on switches instead of the traffic load going through the switches. DCN topology designs, *e.g.*, Fat-tree [2], VL2 [10], BCube [11] provide more network components and more paths between arbitrary pairwise servers. This advantage brings opportunities to improve DCNs energy-proportionality, because turning off a subset of switches would not disconnect servers.

ISBN 978-3-901882-83-8 © 2016 IFIP

While traffic consolidation has been an effective way to achieve energy-efficiency by consolidating flows to fewer switches [13], [19], [21], they heavily depend on the accurate prediction of traffic [7]. On the other hand, most of the traffic engineering based solutions are agnostic to the network flow performance, which results in delaying flows and slowing down responses to requests. This degradation is not acceptable for applications requiring high quality of services (QoS).

An important class of data center applications, called Online Data-Intensive (OLDI) applications, *e.g.*, web search and online retail, employ algorithms where every query operates on data spanning thousands of servers. Latencies in this request-response process would heavily affect users experiences. To avoid the performance degeneration and keeping DCNs energy efficient, we avoid monitoring traffic flows frequently but instead obtain the flows' deadlines implicitly with the TCP AIMD mechanism. Quantitatively, we measure the flow performance in terms of its flow completion time (FCT), as the previous works did [14], [17], [18], [22]. Our objective is to design state changing (on/off) schemes for switches where the DCN energy consumption is minimized and the flow deadlines are met.

In order to reduce a DCN energy consumption, the optimal solution should meet the following goals:

- *Work Conservation* (*WC*): the principle to keep active switches with high utilization.
- *Performance Guarantee* (*PG*): the criterion for energy saving schemes that should not affect network FCTs.
- *Network Agility* (*NA*): the ability to dynamically grow and shrink the DCN capacity to meet traffic loads.

In this paper, we formulate this DCN Energy Saving (DES) problem and prove its NP-completeness. To address the DES problem, we propose a framework to optimize the energy consumptions while maintaining the network performance. To find the most suitable network subset, we present multiple algorithms to select paths for flows such that the flow bandwidth demands are satisfied and the number of required switches is minimized. Each algorithm achieves different tradeoffs between efficiency and optimality. To keep the flow performance, the transmission rates are allocated in a way which meets the flow deadlines and efficiently utilizes the bandwidths. Using flow rate allocation and flow path selection, our framework could dynamically adjust the active set of network elements to satisfy changing traffic loads. Experiments show that compared to existing works, our solution on average, could save 20% of the network energy in data centers, while holding the better flow performance.

The rest of the paper is organized as follows: Section II formulates the DES problem and analyzes its hardness. Section III describes the framework where the rate allocation and path selection algorithms are proposed. Section IV presents the simulation results. We review the related works in Section V and conclude the paper in Section VI.

II. The DCN Energy Saving Problem

In this section, we first analyze the characteristics of DCN switch and the power-down strategy. Based on the preliminaries, we formulate the DES problem as a constrained optimization problem, and analyze its complexity.

A. Preliminaries

A network switch is commonly composed with chassis, line-cards, switching fabric, and ports. The chassis usually consumes a constant power. The line-card buffers packets and the switching fabric maintains the switching table. Ports consume dynamic power according to their speeds. The DES problem critically relies on the nature of the speed-power curve $f(s)$, a function mapping the switch's processing speed s to its power. In [4], [5], [20], the authors calculate $f(s)$ only by switch ports. In this work, $f(s)$ is more general and includes a constant plus a dynamic speed-related component. This model meets the real statistics [16] and supports our idea that, powering down the unused switches can save more energy than speed scaling [4]. Equivalently, we transfer the port power consumption to its associated link power consumption.

Advanced architectures (*e.g.*, Fat-tree [2], VL2 [10], BCube [11]) enrich the connectivities among servers in data centers. But the measurements from [7] show that, the utilization of aggregation switch links is 8% at 90% running time, while the average utilizations of edge layer switches and core layer switches are around 20% and 40%, respectively. This enables us to combine flows from several low utilization switches. As Fig. 1 illustrates, flows could be consolidated onto fewer switches when the DCN is at low utilization. In this example, after flow consolidation, six idle switches could be turned off, reducing network power by nearly 30%.

B. Problem Formulation

We assume that flows $K_1, K_2, ..., K_n$ are transmitted among the DCN. We denote the i-th flow, $K_i = \{s_i,\ t_i,\ d_i\}$, where s_i is the source, t_i is the sink, and d_i is the flow size. The DCN is abstracted as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes, including hosts and switches, and $\mathcal{E}$ is the set of

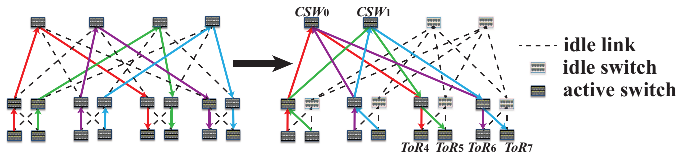


Fig. 1. An example of consolidating flows onto fewer switches in a Fat-tree topology using 4-port switch

links connecting them. Each link $(u,v) \in \mathcal{E}$ has the capacity $c(u,v)$. We do not allow flows to get split due to the fact that the reordering packets would degrade TCP performance, as previous works did [3], [13]. $f_i(u,v)$ is the size of the i-th flow along link (u,v), which is either d_i or 0. An assignment of flows is a mapping from flows to paths in the DCN, such that the following constraints are satisfied.

$$\forall (u,v) \in \mathcal{E}, \sum_{i=1}^{n} f_i(u,v) \le c(u,v) \tag{1}$$

$$\forall i, \sum_{r \in \mathcal{V}} f_i(u,r) = 0, \text{when } u \ne s_i \text{ and } u \ne t_i \tag{2}$$

$$\forall i, \sum_{w \in \mathcal{V}} f_i(s_i,w) = \sum_{w \in \mathcal{V}} f_i(w,t_i) = d_i \tag{3}$$

The constraint (1) restricts the total flows along each link not exceeding the capacity. The flow conservation constraint expressed in (2) means that flows should not be created or destroyed at intermediate nodes. Equality (3) means that the sink receives the same amount of data as that the source sends.

Next, we define the following notations:

- $\mathcal{S}$: the set of all switches;
- P_u: the power of switch u, including the *constant* component plus the *dynamic* component;
- P_u^{cons}: the *constant* power component of switch u;
- $P_{u,v}^{link}(x_{u,v})$: the *dynamic* power component consumed by link (u,v), which is related to the flow rate $x_{u,v}$ on it;
- X_u: the indicator deciding whether the switch u is on;
- $Y_{u,v}$: the indicator deciding whether the link (u,v) is on;
- $R_i(u,v)$: the indicator deciding whether the i-th flow uses link (u,v).

The objective function can be formulated as follows:

$$\min \sum_{u \in \mathcal{S}} X_u \cdot P_u, \tag{4}$$

where

$$P_u = P_u^{cons} + \frac{1}{2} \sum_{v \in \mathcal{S}_u} Y_{u,v} \cdot P_{u,v}^{link}(x_{u,v}), \tag{5}$$

$$x_{u,v} = \sum_{i=1}^{n} f_i(u,v). \tag{6}$$

A factor of $\frac{1}{2}$ in Equation (5) is to eliminate the double counting of each link. For the link power consumption model

$P^{link}_{u,v}(x_{u,v})$, we adopt the power function from [4], a widely used version in the literature:

$$P^{link}_{u,v}(x_{u,v}) = \sigma + \mu x^{\alpha}_{u,v}, \tag{7}$$

where σ, μ and α are constants, and usually $\alpha \geq 1$. To avoid the stability problem incurred by frequently togging on and off links, we assume that a link can be powered off only when it carries no traffic, thus the constraint (1) can be augmented with binary variable $Y_{u,v}$:

$$\forall i, \forall (u,v) \in \mathcal{E}, \sum_{i=1}^{n} f_i(u,v) \leq Y_{u,v} \cdot c(u,v). \tag{8}$$

Actually, when all links from a switch are powered off, the switch can be powered off too:

$$\forall u \in \mathcal{S}, X_u = 1 - \prod_{v \in \mathcal{S}_u} (1 - Y_{u,v}). \tag{9}$$

Since we do not allow flow splitting, we have:

$$\forall i, \forall (u,v) \in \mathcal{E}, f_i(u,v) = R_i(u,v) \cdot d_i. \tag{10}$$

Data center network Energy Saving (DES) Problem: Given flows K_1, K_2, ..., K_n, decide X_u ($\forall u \in \mathcal{S}$), where (4) is minimized and constrains (1)(3)(8)(9)(10) are satisfied.

C. The Hardness of DES Problem

Theorem 1: The DES problem is NP-complete.

Proof: First we consider the adapted-DES problem, where we assume the constant component of the power of switch P^{cons}_u could be evenly partitioned to the active links, and the power of link $P^{link}_{u,v}(x_{u,v}) = \sigma + x_{u,v}$, with $\mu = 1$, $\alpha = 1$, and $\sigma = P^{cons}_u / |\mathcal{S}_u|$. Hence, the objective function of this adapted-DES is:

$$\sum_{(u,v)\in\mathcal{E}} (\sigma + x_{u,v}),$$

which is equal to

$$\sum_{(u,v)\in\mathcal{E}} x_{u,v} + \sum_{u\in\mathcal{S}} P^{cons}_u.$$

The Multi-Commodity Flow (MCF) problem is to minimize

$$\sum_{(u,v)\in\mathcal{E}} a(u,v) \cdot x_{u,v},$$

while satisfying constraints (1)(2)(3). Any MCF problem instance could be reduced to the adapted-DES instance in polynomial time, by letting $a(u,v) = 1$ and adding an additional constant. As the MCF problem is NP-complete for integer flows [8], the adapted-DES problem is NP-complete. Any additional constraints like P^{cons}_u cannot be shared by links or $\alpha > 1$, making the DES problem not easier than the adapted-DES problem. ■

Due to the hardness of DES problem, we explore practical and efficient schemes to improve DCN energy-efficiency and flow performance.

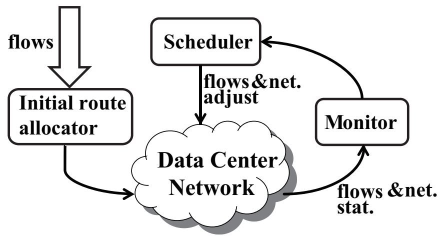


Fig. 2. System framework overview

III. Energy Saving Schemes for DCNs

First we introduce our system framework, as shown in Fig. 2. Typically, flows are initialized by initial route allocator. As flows arrive and depart, the monitor periodically detects the D-CN configurations, and when the network gets suboptimal, the scheduler adjusts flow routes and the network configurations. In this adjusting process, we decouple rate[1] allocation and path selection for flows, aiming at satisfying flow rate demands and minimizing the occupied network simultaneously. For rate allocation, we first apply a simple method to label *implicit* deadlines to deadline-agnostic flows, and use them to calculate the bandwidth demands (III.*A*). Based on the bandwidth demands, linear programming and simulated annealing methods are proposed to select flow paths, and to output the minimum subset of the DCN (III.*B*). The actual rate that a flow obtains is the minimum allocated bandwidth along the selected path. We also extend the methods for practical considerations (III.*C*).

A. Flow Rate Allocation

1) Bandwidth demand calculation: One of the main metrics of the flow performance is its duration time, in other words, the flow completion time. For applications in data centers, especially OLDI applications, flows need to meet their *deadlines* to be useful. Nevertheless, deadlines are hard to explicitly acquire from the packet headers. Since the *sizes* of flows are easy to obtain, we can exploit the sizes of flows to estimate the implicit deadlines for flows, and then use them in bandwidth allocations. The method of calculating deadlines is based on the TCP AIMD mechanism.

When the network is lightly loaded, the TCP (Reno version) congestion window shows the sawtooth wave shape (Fig. 3). The area between the wave and the x-axis can be considered as the amount of data a flow sends. In a sawtooth wave (the gray region), the amount of data is around $\frac{3}{4}WL$ ($L \cdot \frac{W}{2} + \frac{1}{2} \cdot L \cdot \frac{W}{2}$). We assume that the window size starts from $\frac{W}{2}$, and increases linearly until congestion occurs. Given the flow size A, it is easy to approximate the flow duration time by a simple formula:

$$D = \frac{A}{\frac{3}{4}WL} \cdot L = \frac{4A}{3W}.$$

We regard D as the implicit deadline, and this value would be tighter than the time it really takes when transmitting in DCNs, due to the assumption on light network traffic. Given

[1]We use the term rate and bandwidth interchangeably.

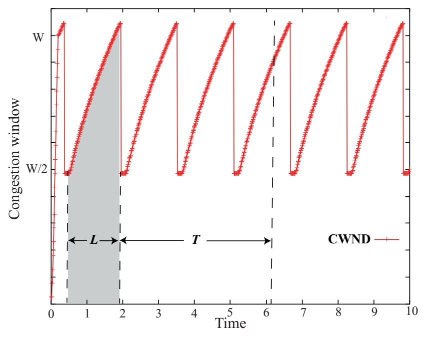


Fig. 3. TCP congestion window

a flow with remaining size $f_i.a$ and the remaining time until its deadline expires $f_i.r$, the demand bandwidth is estimated as $f_i.de = \frac{f_i.a}{f_i.r}$.

2) Rate allocation: The bandwidth demand request travels along the selected path to its destination, along which the nodes allocate bandwidths accordingly. The actual bandwidth flow f_i gets is the minimum of the allocated bandwidths. We apply the Greedy Bandwidth Allocation (GBA, Algorithm 1) for each node, which receives bandwidth requests from flows. If the node has sufficient capacity, each flow will acquire bandwidth $f_i.al = f_i.de + b$, where b is the spare bandwidth capacity shared with other flows. When the node does not have enough capacity to satisfy all the requests, GBA will try to satisfy requests as many as possible. The way we calculate $f_i.al$ respects *PG* goal and outperforms original TCP in terms of FCTs, as shown in Section IV.

Algorithm 1 Greedy Bandwidth Allocation

```
Input: $f[1...N]$: flows to be allocated, B: bandwidth capacity
Output: flows $f[1...N]$ with bandwidth allocations
1: Sort $f[1...N]$ by $f.de$ in a non-decreasing order;
2: $i = 1$, $R = \mathbf{B}$;
3: while $i \leq N$ and $R > 0$ do
4:     $f_i.al = \mathbf{min}\{R, f_i.de\}$;
5:     $R = R - f_i.al$;
6: end while
7: if $R > 0$ then
8:     $b = R/N$, $i = 1$;
9:     while $i \leq N$ do
10:        $f_i.al = f_i.al + b$;
11:    end while
12: end if
13: Return $f[1...N]$;
```

B. Flow Path Selection

Based on the flow bandwidth demands, we have developed two methods to compute a minimum-power network subset in DCNs. The first is based on linear programming, with a randomized rounding scheme for minimizing active switches. Inspired by [3], the second method uses simulated annealing technique to get the solution by efficiently searching solution space. Each method includes both initial route allocator and scheduler algorithms.

1) Linear programming method: First we introduce our initial route allocator in this method. Energy-Efficient Routing (EER, Algorithm 2) counts the number of active switches in each path and sorts the paths in a non-increasing order. Next, EER checks the remaining bandwidths of the ordered paths ($p_i.re$) successively and allocates them to the candidate flow until satisfying its demand (*f.de*). The divisor (Λ − N[i]) (line 13) is small when there are many active switches in this path, and in this case the probability of choosing this path is high . We set $\Lambda = 6$ in Fat-tree, since there are 5 switches between inter-pod hosts. For example, $f.de = 20$, N[1] = 5, N[2] = 2, A[1] = 10 and A[2] = 10, after *Normalize*(), the probability of choosing p_1 and the probability of choosing p_2 are 0.8 and 0.2, respectively. EER tends to allocate flows to paths which have more active switches and to meet *WC* goal.

Algorithm 2 Energy-Efficient Routing

```
Input: $f$: flow to be allocated, $p$: the set of $M$ possible paths
Output: an ideal path chosen for $f$
1: for $i$: 1 to $M$ do
2:     N[$i$] = CountActiveSW($p_i$);
3: end for
4: Sort $p$ by number of active switches in decreasing order;
5: for $i$: 1 to $M$ do
6:     if $p_i.re \leq 0$ or $f.de \leq 0$ then
7:         continue;
8:     end if
9:     N[$i$] = CountActiveSW($p_i$);
10:    A[$i$] = min$\{p_i.re, f.de\}$;
11:    $p_i.re = p_i.re -$ A[$i$];
12:    $f.de = f.de -$ A[$i$];
13:    A[$i$] = A[$i$] / ($\Lambda -$ N[$i$]);
14: end for
15: Normalize(A[1...M]);
16: Return $p_i$ with probability A[$i$];
```

As flows arrive and depart, the network utilization may become suboptimal. The scheduler takes the network configuration and the flows as inputs and recomputes the paths for flows in nearly real-time, such that the active switches are minimized and the *NA* goal is respected. For scheduler algorithm in this method (LP, Algorithm 3), LP solves the MCF problem with fractional linear programming, and then extracts candidate paths for each flow. With the fractional results, we use a natural randomized rounding scheme, where the paths are chosen with the probabilities depending on their weights. Similarly to EER, we measure the weight of the j-th candidate path for the i-th flow by the assigned bandwidths $\mathcal{P}_i[j].as$ and the number of active switches N[j], giving that LP respects *WC*.

2) Simulated annealing method: Directly computing the flow assignment needs exhaustive search in the solution space, which is exponential to the number of flows. We introduce

Algorithm 3 Linear Programming Based Path Selector

```
Input: $f[1...N]$: flows to be allocated, $\mathcal{G}$ = ($\mathcal{E}$,$\mathcal{V}$): the DCN
Output: routing path $p_i$ for $f[i]$
1: Solve MCF problem by linear programming;
2: For each flow $f[i]$, extract a set of candidate paths $\mathcal{P}_i$;
3: for $i$ : 1 to $N$ do
4:     for $\mathcal{P}_i[j] \in \mathcal{P}_i$ do
5:         N[$j$] = CountActiveSW($\mathcal{P}_i[j]$);
6:         $w[i][j]$ = $\mathcal{P}_i[j].as$ / ($\Lambda-$N[$j$]);
7:     end for
8:     Normalize($w[i]$);
9:     Choose $p_i$ with probability $w[i]$;
10: end for
11: Return $p_1$, $p_2$, ..., $p_N$;
```

a novel method which can significantly reduce the solution space. The key insight is that a core switch can handle multiple flows destined to specific hosts. In Fig. 1, *CSW*$_0$ handles flows destined to hosts under *ToR*$_4$ and *ToR*$_6$, while *CSW*$_1$ handles flows destined to hosts under *ToR*$_5$ and *ToR*$_7$. That is, we shift from choosing paths for flows to choosing core switches for hosts. Once the mapping from hosts to core switches is determined, so are the paths for flows. Both initial route allocator and scheduler algorithm in this method depend on this mapping, since initial route allocator could initialize paths for flows by simply searching in this mapping in terms of their destinations, and scheduler updates this mapping by calling SA (Algorithm 4).

SA is to find the optimal state in the state space, where a state is a particular mapping. In each iteration, we generate a neighboring solution from current state and accept it as the new state with a probability depending on the energy of current and neighboring state, and current temperature. SA proceeds the iterations, with a decreased temperature each round, and it stops when the temperature reaches 0.

Algorithm 4 Simulated Annealing Based Path Selector

```
Input: $s$: initial state, $n$: iteration count
Output: $s_B$: beat state of the mapping
1: $e$ = Energy($s$);
2: $s_B$ = $s$, $e_B$ = $e$;
3: for $T$: $n$ to 0 do
4:     $s_N$ = Neighbor($s$);
5:     $e_N$ = Energy($s$);
6:     if $e_N < e_B$ then
7:         $s_B$ = $s_N$, $e_B$ = $e_N$;
8:     end if
9:     if Pr($e_N-e$, $T$) $>$ Rand() then
10:        $s$ = $s_N$, $e$ = $e_N$;
11:    end if
12: end for
```

With the flow assignment determined by SA, we could easily find whether the flow demands in a link exceed its capacity. The *Energy*() function is defined by the total exceeded bandwidth demands and the number of active switches in *s*:

$$Energy(s) = (s.ex\_band + 1) \cdot (s.active\_sw + 1).$$

An extra 1 is necessary to ensure that the energy is always positive and comparable. *Pr*() defines the probability of accepting neighboring state as the new state:

$$Pr(\Delta E, T) = \begin{cases} 1 & \Delta E > 0 \\ e^{c\Delta E/T} & \Delta E \leq 0 \end{cases}$$

where c is a adjustable parameter.

Initial state: Particularly, the number of core switches is equal to the number of hosts in a pod in the Fat-tree. We restrict our first initial state to one-to-one mapping, which implies each host in a pod is mapped to a unique core switch. In subsequent scheduling phases, we set the initial state as the best state from the previous phase. This configuration could reduce the disruption of existing flows.

Neighbor state generator: Our neighbor state generator directs SA to appropriate mappings which saves more energy and tries to meet the capacity constraints. Our strategies are: (i) powering off a randomly-chosen core switch and remapping the involved hosts to other randomly-chosen core switches, (ii) swapping the mapping relationships of two randomly-chosen hosts, (iii) remapping a randomly-chosen host to another core switch, and (iv) powering on a core switch and remapping a randomly-chosen host from other switches to this switch. The first strategy is for energy-saving, while the last three strategies are for fine placements of flows with less exceeded capacity bandwidths. These four strategies are chosen with equal probability.

Energy function: The energy function we define involves both energy and bandwidth allocation information. Less energy indicates either less energy consumptions or less exceeded bandwidths, either of which would be accepted as a *good* state. Given a fixed traffic, less energy consumption mean less active switches, and then higher switch utilizations, giving that SA respects *WC*.

C. Practical Considerations

Stability consideration: The algorithms we have proposed are to find the minimum subset of switches, which achieve the three goals presented in Section I. However, these may go too far and result in unstable load. Unstable load will lead to different subsets of switches in the contiguous scheduling phases, meaning that the states of several switches change frequently. For stability consideration, we extend our algorithms with a *Hit mechanism.*

After applying the scheduler algorithm (LP or SA), we extract the path for each flow, with which the subset of switches is determined to run until next schedule phase. For a switch not in the subset, (*i.e.*, the switch would be turned off), if the number of requests for powering off this switch in the scheduling interval is greater than a given *threshold*, we actually power off this switch, otherwise, we keep the switch on and randomly choose flows from geographically neighbor switches to adjust their paths to go through this switch. We correspondingly call the algorithms with *Hit mechanism* as LP-hit and SA-hit.

Fault tolerance consideration: The framework could certainly minimize the network energy consumption, but may hurt the performance regarding fault tolerance of the original system that always keeps switches on. We refer to the original system as *always-on baseline* in our later evaluation. In current data centers, failures are quite common due to hardware, software, and power outage problems [6], [12]. We thus improve our algorithms for fault tolerance with minor modifications. For LP, we modify the *Normalize*() (Algorithm 2, line 8) function. For example, assume that there are two possible candidate paths for flow 0, whose weights are $w[0][0]$ = 10, and $w[0][1]$ = 2.5, respectively, and other candidate paths are with weight $w[0][i]$ = 0, for $i \geq 2$. In LP-FT, we make the *impossible* paths join the candidate path set by modifying their weights, (in this example, let $w[0][2]$ = 2). This leads to the probabilities of choosing path 0, 1, and 2 are 69.0%, 17.2% and 13.8%, respectively. This method activates more paths, and powers on more switches for fault tolerance. For SA-FT, we let additional new core switches join the *best* mapping computed by SA, and change a randomly-chosen host to map to the new core switch.

Our framework combines GBA with LP and SA, respectively. The LP based method and its variants run fast, while the SA method and its variants achieve more energy savings and maintain better FCTs, as demonstrated in the next section.

IV. Simulation Evaluations

This section describes evaluations of our system framework. The goal of these tests is to determine the energy-efficiency and flow performance under various traffic patterns and various network utilizations. The efficiencies and fault-tolerance of algorithms are also evaluated.

A. Simulation Setups

We simulated our system framework on a laptop with an Intel Core i5-2410M 2.30Ghz CPU and 4GB RAM. All of the algorithms are implemented in Java.

Our simulator captures the flow-level events like flow arrivals, departures and transmission rate calculations. Existing packet-level simulators such as *ns*-2 become extremely slow or even impossible when the number of network nodes becomes huge. The traffic is generated for a range of communication patterns at the granularity of network flow which follows Pareto distribution with mean size 50KB. In our simulations, time is split into slices. At each time slice, it updates flow rates and generates new flows if needed. Periodically it calls the scheduler to reassign flows to new paths and changes switch states. For comparison, we also implement the *Greedy Bin Packing* (GBP) method [13], which evaluates possible paths and chooses the leftmost one with sufficient capacity for each flow. When calling the scheduler algorithm (LP, SA or GBP), the simulator also calls GBA for bandwidth allocation. When updating flow rates, we also implement the TCP with slow start and AIMD, and D3 [22] for flow performance comparisons. Our simulator has similar implementation as that in [3], which matches testbed performance very well.

We use Fat-tree topology with 320 switches and 1024 servers. Similar to previous works [2], [3], our synthetic traffic patterns include:

- *Random*: Each server sends to a random destination. Multiple servers can send to the same receiver.
- *Random bijection*: Each server sends to a random destination. Each host receives data from only one sender.
- *Random nonpod*: Each server sends to a random destination not in the same pod as itself. Multiple servers can send to the same receiver.
- *Stride*-512: We number the servers in our Fat-tree topology from left to right, as the leftmost is 0 and the rightmost is 1023. By Stride-512, we mean the server i sends to server (i+512) $\bmod$ 1024.

For a single switch power function (Equation (5)), we set P_u^{cons} = 200 watts, and for each link power function σ = 0, μ = 2 × 10^{-6} watts/(Mbps)^2 and α = 2, which are adopted from [19]. Consequently, the maximum power consumption of each switch is 232 watts.

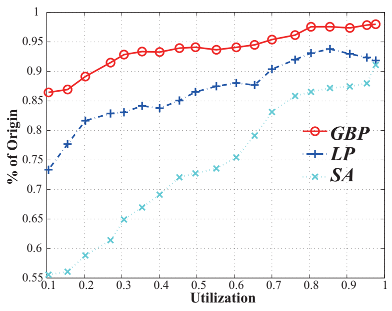
B. Simulation Results

We now explore the efficiency of our framework. The primary metrics include (i) FCTs, and (ii) the ratio of energy consumption with our algorithms over the energy consumption with the always-on baseline.

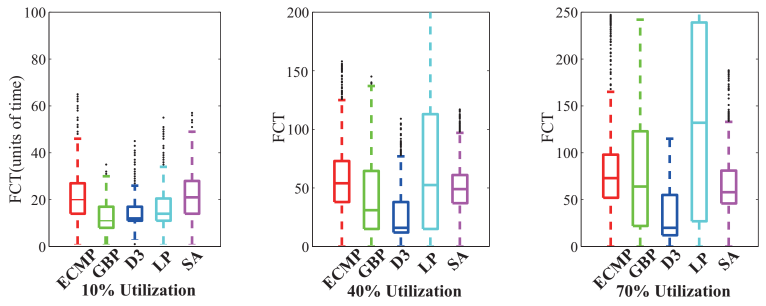
1) Synthetic demands, varying loads: Energy savings and flow performance heavily depend on the traffic patterns and network utilizations.

Energy saving evaluations: Figs. 4(a), (c), (e), (g) show the energy savings in the Random, Random bijection, Random nonpod, Stride-512 traffic patterns, respectively. In each traffic pattern, we vary the network utilization from 10% to nearly 100% by adding more flows. And for each utilization and each traffic pattern, we run the simulation for 60 seconds, and measure the average energy consumptions during the middle 40 seconds. In all four traffic patterns, SA saves more energy than GBP and LP. In detail, 25% and 8% more energy (at least) can be saved by SA compared with GBP and LP model at low utilization (10% $-$ 30%), respectively. While the gaps in energy saving between different algorithms decrease as the utilizations grows, SA could save 18% and 7% (on average) more energy than GBP and LP model at medium utilization (30% $-$ 60%), respectively. When the utilization is close to 100%, all the switches must remain active, and thus all algorithms have similar performance. From another point of view, traffic patterns also affect energy savings. At 20% utilization, SA achieves 42% and 35% energy savings in Random and Random nonpod traffic patterns, respectively, implying that at the same network utilization, the less flows through core switches, the more energy saving.

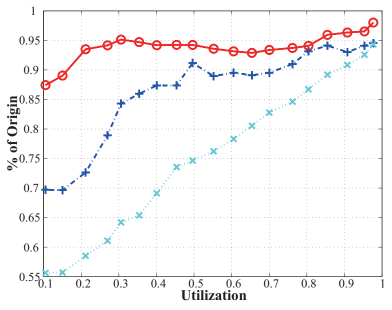
Flow performance evaluations: In this section, we evaluate the flow performance of different algorithms under synthetic



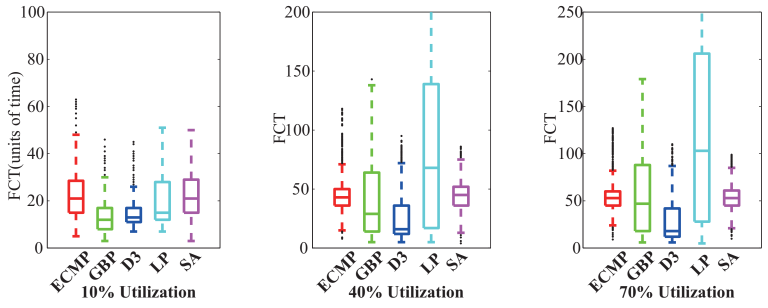
(a) Energy under Random traffic



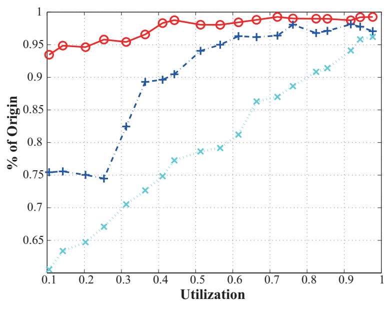
(b) FCTs under Random traffic



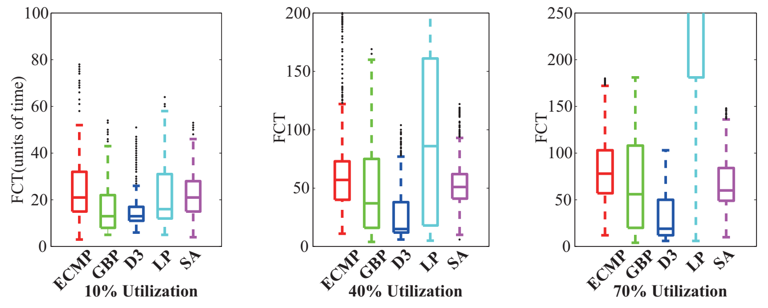
(c) Energy under Random bijection traffic



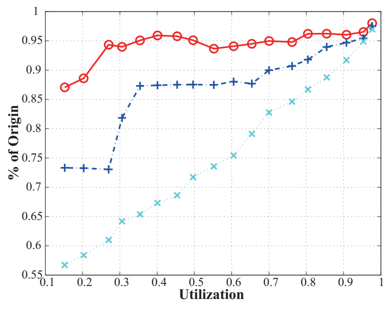
(d) FCTs under Random bijection traffic



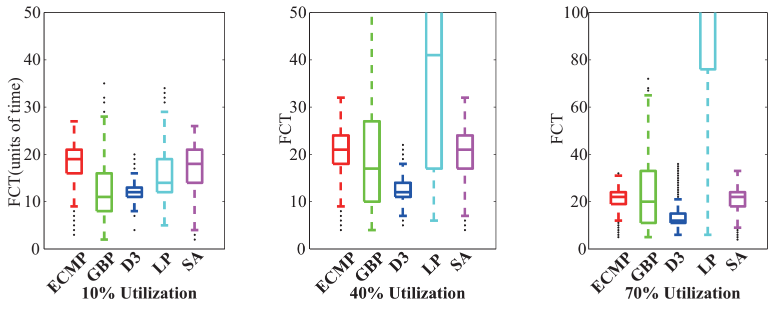
(e) Energy under Random nopod traffic



(f) FCTs under Random nopod traffic



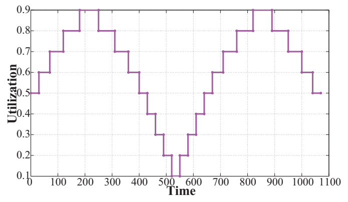
(g) Energy under Stride-512 traffic



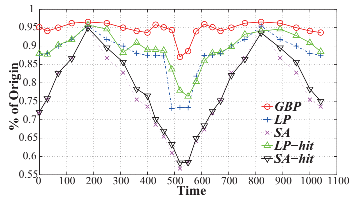
(h) FCTs under Stride-512 traffic

Fig. 4. Energy consumptions and FCTs under synthetic traffic patterns

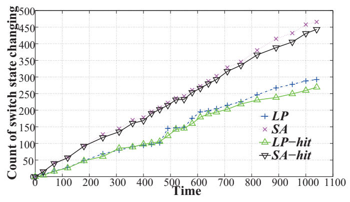
traffic patterns. Here, we choose three utilizations of 10%, 40% and 70% to represent low, medium and high utilizations, respectively, and focus on the FCTs. The ECMP and D3 schemes run within the always-on network configuration, while SA, LP and GBP run within subsets of the network. Figs. 4(b), (d), (f) and (h) plot the percentiles FCTs (1^{st}-25^{th}-50^{th}-75^{th}-99^{th}) at different traffic patterns and network utilizations. At low utilization, the median FCTs for all approaches are comparable, while for the 99^{th} percentile FCT, SA and GBP are comparable, but LP performs worse than the two algorithms. As network utilization increases, LP model becomes worse than SA and GBP, because its probabilistic path selection leads to aggressive flow congestions. At medium load, the median FCT of SA is, on average, 11% higher than that of GBP, while the 99^{th} percentile of SA is at least 33% lower than that of GBP. The gaps between SA and GBP



(a) Utilization



(b) Energy consumptions



(c) Count of switch state changing

Fig. 5. Energy consumptions with sine-wave utilization

TABLE I
THE RATIOS OF ENERGY CONSUMPTION WITH SA BY VARYING ITERATIONS

SA Iteration	Random			Random bijection			Random nopod			Stride-512		
	10%	40%	70%	10%	40%	70%	10%	40%	70%	10%	40%	70%
50	0.594	0.726	0.860	0.581	0.720	0.828	0.618	0.781	0.913	0.557	0.681	0.856
100	0.570	0.703	0.840	0.562	0.703	**0.819**	0.609	0.766	0.887	0.557	0.673	0.832
500	0.573	0.695	0.831	0.560	0.708	0.820	0.605	0.770	0.878	**0.550**	0.690	0.827
1000	**0.563**	0.702	0.838	0.554	0.696	0.822	0.610	0.765	0.891	0.564	**0.668**	0.831
5000	0.567	**0.691**	0.834	0.562	0.695	0.821	**0.604**	0.772	0.875	0.551	0.678	0.823
10000	0.565	0.695	**0.830**	**0.551**	**0.691**	0.825	0.605	**0.761**	**0.871**	0.560	0.670	**0.818**

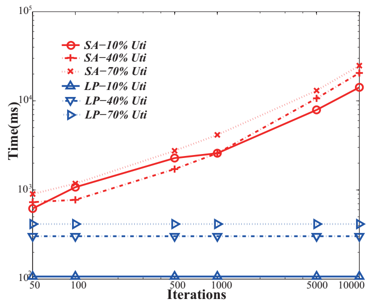
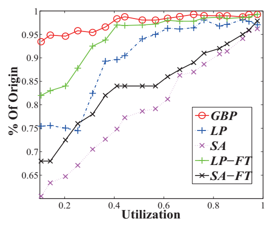
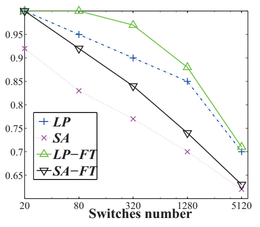


Fig. 6. Time consumptions by varying iterations



(a) Varying utilizations



(b) Varying scales

Fig. 7. Energy consumptions with fault tolerance

become larger in terms of the 99th percentile, but become closer in terms of the median at high utilization.

2) Diurnal variation demand: As the network utilization varies with time in product data centers, we capture this demand by simulating a sine-wave utilization which is inspired by [13]. As Fig. 5(a) shows, the highest utilization is 90%, while the lowest is 10%. The simulation runs for 40 seconds and 80 seconds at low and high utilization, respectively, to imitate the diurnal utilization pattern. We compare energy-efficiency of LP and SA, together with LP-hit and SA-hit. From measurements of the half of the wave length, we record and calculate the energy consumptions from the 200th time tick to the 550th time tick, which demonstrate that SA and LP save 18.4% and 5.6% more energy than GBP, respectively. While SA-hit consumes more energy than SA, which is about 3% in this load decreasing phase, it consumes almost equal energy as SA in the load increasing phase. This phenomenon also appears in LP-hit, as illustrated in Fig. 5(b). Since *Hit mechanism* indeed incurs more energy, it could reduce the number of switch state changing (Fig. 5(c)). We also note that SA leads to more state changing times than LP, which we believe, is the expense for higher energy savings.

3) Iterations and running times: To explore the energy savings of SA under different iterations, we conduct the experiments using synthetic traffic patterns and various utilizations as before. These results (Table I) conform with our expectation that more iterations create *better* state that can be used to save more energy. Fortunately, most of the performance improvements appear in the first few iterations. Next, Fig. 6 shows the running times of SA for different iterations, compared with LP under various utilizations. These results report that the running times of SA with few hundreds of iterations are not much longer than LP.

4) Fault tolerance: We use the Fat-tree topology as before, and the Random traffic pattern as the underlying traffic. For LP-FT, we add two more possible paths into the candidate path set for each flow, while adding one more core switch (if exists) for each pod to the best state for SA-FT. Fig. 7(a) shows that additional energy required by fault tolerance decreases as the utilization grows. Furthermore, as the network scale increases, the cost of fault tolerance decreases, which is confirmed by the results of Fig. 7(b) where the number of switches changes from 20 to 5120.

In summary, our SA and LP methods together with GBA show better energy-efficiency than GBP algorithm under the synthetic traffic patterns. In terms of flow performance, LP impacts the FCT aggressively. In contrast, SA shows comparable median FCTs but lower 99th percentiles compared with GBP. Besides, the running time of SA is not much longer than LP since its improvements almost always appear in the first few iterations. We also demonstrate that our methods can be easily extended for fault tolerance and the incurred energy consumptions are controllable.

V. Related Works

Energy-efficient DCNs: Many approaches have been proposed on improving the energy-efficiency of DCNs. The first type of these works is to design novel architecture. [15] proposed a server-centeric data center structure that conserves energy by varying bandwidth availability based on traffic demand. The second type is to optimize energy-efficiency of DCNs by traffic engineering methods. Heller *et al* [13] presented *ElasticTree*, a network-wide power manager, which dynamically adjusted the set of active network elements. But *ElasticTree* assumes that a complete prior knowledge of incoming traffic is known. Wang *et al* [21] proposed CARPO, a correlation-aware power optimization algorithm that dynamically consolidates traffic flows onto a set of switches and shuts down unused network devices. Andrews *et al* exploited speed scaling [4] and power-down [5] techniques to route and schedule continuous flows, but the transmission speed for each flow was given as a constant.

Flow-level optimization: The performance of OLDI applications heavily depends on FCTs. There is abundant work that deals with the subject of data center transport designs. D3 [22] first introduced deadline information combined with explicit rate control. PDQ [14] showed that minimizing FCTs requires preemptive flow scheduling.

There are few works that improve DCN energy-efficiency and performance simultaneously. Wang *et al* [20] proposed a novel energy-saving model for data center networks by scheduling and routing deadline-constrained flows, based on speed scaling and power-down strategies. But the power function of switches they used was only based on links (or ports). We use the general switch power function and propose approaches that could make energy consumptions of DCNs approximately in proportion to their traffic loads, while the flow-level performance degradation is guaranteed.

VI. Conclusion

In this paper, we studied the energy-efficiency of DCNs, where the flow-level performance was guaranteed. The key idea of our framework was that we decoupled the path selection and rate allocation for flows. In path selection, we assigned flows to paths with fewer switches to minimize network energy consumptions. In rate allocation, we efficiently utilized link bandwidth to satisfy flow demands which were estimated with implicit deadlines. With these approaches, we were able to achieve both energy efficiency and better flow performance compared with existing works.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China under Grant No. 61472181, 61321491, 61502224, 91218302; China Postdoc Science Fund under Grant No. 2015M570434; Jiangsu Natural Science Foundation under Grant No. BK20151392; Jiangsu Key Technique Project (industry) under Grant No. BE2013116; EU FP7 IRSES MobileCloud Project under Grant No. 612212. And this work is also partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] America's data centers consuming and wasting growing amounts of energy. http://www.nrdc.org/energy/data-center-efficiency-assessment.asp.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. *ACM SIGCOMM 2008*, pages 63–74.

[3] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *NSDI 2010*, pages 19–19.

[4] M. Andrews, A. F. Anta, and L. Zhang. Routing for power minimization in the speed scaling model. *IEEE/ACM TON*, 20(1):285–294, 2012.

[5] M. Andrews, A. Fernández Anta, L. Zhang, and W. Zhao. Routing and scheduling for energy and delay minimization in the powerdown model. *Networks*, 61(3):226–237, 2013.

[6] L. Barroso, J. Dean, and U. Holzle. Web search for a planet: The google cluster architecture. *Micro, IEEE*, 23(2):22–28, March 2003.

[7] T. Benson, A. Anand, A. Akella, and M. Zhang. Understanding data center traffic characteristics. *ACM SIGCOMM 2010*, pages 92–99.

[8] S. Even, A. Itai, and A. Shamir. On the complexity of time table and multi-commodity flow problems. In *FOCS 1975*, pages 184–193.

[9] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM 2008*, pages 68–73.

[10] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. Vl2: a scalable and flexible data center network. In *ACM SIGCOMM 2009*, pages 51–62.

[11] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. Bcube: a high performance, server-centric network architecture for modular data centers. *ACM SIGCOMM 2009*, pages 63–74.

[12] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. Dcell: A scalable and fault-tolerant network structure for data centers. In *ACM SIGCOMM 2008*, pages 75–86.

[13] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown. Elastictree: Saving energy in data center networks. In *NSDI 2010*, pages 249–264.

[14] C.-Y. Hong, M. Caesar, and P. Godfrey. Finishing flows quickly with preemptive scheduling. *ACM SIGCOMM 2012*, pages 127–138.

[15] L. Huang, Q. Jia, X. Wang, S. Yang, and B. Li. Pcube: Improving power efficiency in data center networks. In *CLOUD 2011*, pages 65–72.

[16] P. Mahadevan and P. Sharma. A power benchmarking framework for network devices. In *NETWORKING 2009*, pages 795–808.

[17] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan. Minimizing flow completion times in data centers. In *INFOCOM 2013*, pages 2157–2165.

[18] B. Vamanan, J. Hasan, and T. Vijaykumar. Deadline-aware datacenter tcp (d2tcp). *ACM SIGCOMM 2012*, pages 115–126.

[19] L. Wang, F. Zhang, J. Arjona Aroca, A. V. Vasilakos, K. Zheng, C. Hou, D. Li, and Z. Liu. Greendcn: a general framework for achieving energy efficiency in data center networks. *IEEE JASC*, 32(1):4–15, 2014.

[20] L. Wang, F. Zhang, K. Zheng, A. V. Vasilakos, S. Ren, and Z. Liu. Energy-efficient flow scheduling and routing with hard deadlines in data center networks. In *ICDCS 2014*, pages 248–257.

[21] X. Wang, Y. Yao, X. Wang, K. Lu, and Q. Cao. Carpo: Correlation-aware power optimization in data center networks. In *INFOCOM 2012*, pages 1125–1133.

[22] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron. Better never than late: Meeting deadlines in datacenter networks. In *ACM SIGCOMM 2011*, pages 50–61.