Greening Data Center Networks with Throughput-guaranteed Power-aware Routing

Mingwei Xu[a,b], Yunfei Shang[a,b], Dan Li[a,b,*], Xin Wang[d]

[a]*Department of Computer Science and Technology, Tsinghua University, Beijing, P.R. China*

[b]*Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing, P.R. China*

[d]*Department of Electrical and Computer Engineering, State Univ. of New York at Stony Brook, NY, United States*

Abstract

Cloud based applications and services require high performance and strong reliability provided by data center networks. To overcome the problem of traditional tree based data center network, recently many new network architectures are proposed, such as Fat-Tree and BCube. They use aggressively over-provisioned network devices and links to achieve 1:1 oversubscription ratio. However, most of the time data center traffic is far below the peak value and a large number of idle network devices and links in data centers consume a significant amount of power, which is now becoming a big problem for many cloud providers.

In this paper, we aim to reduce the power consumption of high-density data center networks from the routing perspective while meeting the network performance requirement. We call this kind of routing *throughput-guaranteed power-aware routing*. The essence of our idea is to use as little network power as possible to provide the routing service, without significantly compromise on the network performance. The idle network devices and links can be shut down or put into the sleep mode for power saving. Extensive simulations conducted in typical data center networks show that our power-aware routing can effectively reduce the power consumption of network devices, especially under low network loads.

Keywords: Throughput-guaranteed power-aware routing; data center network; network power consumption model.

1. Introduction

Today's data centers integrate a great number of switches and servers to provide various cloud-based services, such as online search, web mail, e-business, as well as basic computational and storage functions, such as MapReduce [1], GFS [2], and CloudStore [3]. The goal of the data center network (DCN) is to interconnect the massive number of data center servers, and provide efficient and fault-tolerant routing to support upper-layer applications. It has attracted great attention to design a reliable and efficient DCN architecture recently. The traditional tree architecture applied in current data centers [34] is known to face many challenges in supporting bandwidth-hungry communications in data centers. More specifically, the tree structure suffers from low

*Corresponding author. Tel/fax: 86-010-62603064/86-010-62603062.
Email address: xmw@csnet1.cs.tsinghua.edu.cn (Mingwei Xu), shangyunfei@csnet1.cs.tsinghua.edu.cn (Yunfei Shang), lidan@csnet1.cs.tsinghua.edu.cn (Dan Li), xwang@ece.sunysb.edu (Xin Wang)
This work was supported by HI-Tech Research and Development Program of China (863) under Grants No. 2007AA01Z2A2, NSFC under Grants No. 61073166 and 973 Program under Grants No. 2009CB320502.
Some preliminary results of this work appeared in ACM SIGCOMM Workshop on Green Networking 2010.

scalability, high cost as well as single point of failure. Hence, recently many advanced network architectures are proposed to mitigate these issues, represented by Fat-Tree [4], BCube [5], etc. These new data center architectures use more network devices and links to effectively overcome the shortcomings of the tree architecture and to enjoy 1:1 oversubscription ratio.

In order to better support data-intensive applications in data centers, these "richly-connected" network architectures are designed with the major purpose of ensuring high communication performance and robustness. These architectures have two characteristics in common: *over-provisioned network resources* and *inefficient power usage*. An excessive number of network devices and redundant links are provided aggressively for the busy-hour load. However, most of the time, the traffic in a data center is far below the peak value and varies greatly between daytime and night, which leaves a large number of network devices to stay idle. The goal of network power conservation is to make the power consumption on networking devices proportional to the traffic load [35]. In existing data centers, however, the network at the low load still consumes more than 90% of power used at the busy-hour load [7]. So a large number of idle network devices in high-density networks consume a significant amount of power, which results in an extremely inefficient power usage in data center networks.

The power cost brought by hardware devices such as network devices and servers in data centers accounts for a dominant part of the operational costs of data centers, and it may skyrocket as the scale of data centers expands. The power cost is becoming a big burden for many cloud providers. According to the statistics, the total power consumption of global data centers accounts for 1.1-1.5% of the worldwide electricity use in 2011 [8], and the figure will continually increase to 8% by 2020 under the current trend [42]. It has been shown that network devices consume about 20% power in the whole data center [7], and the ratio will grow with the rapid development of power-efficient hardware and power-aware scheduling algorithms on the server side. Therefore, it is of high importance to investigate advanced power conservation technologies for data center networks, which will in turn bring a great benefit in reducing the operational cost of data centers and contribute to the reduction of the carbon footprint.

The objective of this paper is to propose a novel *throughput-guaranteed power-aware routing* algorithm to reduce the total power consumption of network devices, and to make power usage more efficient in "richly-connected" data center networks. The key idea is to use as little network power as possible to provide the routing service, while maintaining the target network throughput. The idle network devices and links can be shut down or put into sleep for power saving. We also consider the tradeoff between power conservation and network fault-tolerance. Our algorithm can flexibly adapt the power-aware routing to meet different reliability requirements. In contrast to previous power-aware routing work, our approach uses a different traffic rate model, in which the transfer rate of each flow is bandwidth-constrained and depends on the network resource competition with other flows, instead of given by the fixed traffic demand in advance. This is because network is becoming the bottleneck for data-intensive distributed computation in data centers.

We make the following contributions in the paper. First, we formally establish the model of throughput-guaranteed power-aware routing problem, which will guide us to effectively analyze and solve the problem. We analyze the time complexity of the power-aware routing problem, and prove that it is NP-hard (Section 2.2 and Appendix).

Second, we propose a throughput-guaranteed power-aware routing algorithm to achieve our design goal (Section 3). The algorithm works in the following four steps: Step 1, we compute the routing paths and corresponding network throughput with all switches and links in the initial topology of the data center network, which are called *basic routing* and *basic throughput* respectively. Step 2, we introduce an iteration process to gradually remove switches from the basic routing and update the initial topology, while satisfying the predefined performance requirement. Step 3, we remove as many links connected to active switches as possible from the updated topology above while meeting the throughput requirement. Step 4, we further adapt the updated topology to meet the reliability requirement. We can power off the switches and links not involved in the finally updated topology, or put them into the sleep mode to conserve power.

Third, we conduct extensive simulations in typical data center networks to validate the effectiveness of our power-aware routing algorithm under different power-saving granularities, traffic patterns and reliability requirement (Section 4). The results show that our power-aware routing algorithm can effectively reduce the power consumption of network devices in data center networks, especially under low network loads.

The rest of the paper is organized as follows. Section 2 introduces background knowledge and related work, and formally establishes the throughput-guaranteed power-aware routing problem model. Section 3 presents our algorithm design. Section 4 evaluates the algorithm through simulations in typical data center networks. Section 5 discusses practical implementation issues of our algorithm and Section 6 concludes the paper.

2. Background and Model

In this section, we first introduce the power consumption models as well as the power conservation strategies of network devices in data centers, and then establish the model of throughput-guaranteed power-aware routing problem. Finally, we present the related work on green Internet and data center networks.

2.1. Network Power Consumption Model

We discuss two types of power consumption models applied to current modular network devices in data centers. The first one, called *General Model*, can accurately calculate the power consumption of network devices, but its dependence on unpredictable network traffic conditions increases the complexity of the computation and thus the model is hardly used in practice. To acquire a practical model, we discuss the *Simplified Model* which is a more feasible model. It can easily compute the power consumption of switches only based on their configurations, regardless of the factor of the network traffic. Note that we are not claiming any novelty in the power consumption model of network devices, but discuss here to make the paper more complete. We will use the network power consumption model to guide the design and evaluate the effectiveness of the throughput-guaranteed power-aware routing algorithm in Sections 3 and 4 respectively.

General Model

The power consumption of modular network devices depends on their configurations and the characteristics of traffic passing through them, according to the power benchmarking results of various network devices in [28]. Thus, we take these two types of parameters as the general model's inputs. The configuration parameters of network devices contain: the type of chassis and linecards, the number of linecards and ports, the port capacity, etc. In terms of traffic characteristics, the authors in [28] have found that the power consumption of network devices may be affected by the rate of traffic across them, instead of packet size and other traffic characteristics.

The power consumption of a general modular switch or router can be divided into three main parts [20], [29] in the equation (1). $P(C,T)$ denotes the total power consumption of a network device. C and T are the input parameters, representing the configuration of the network device and the characteristics of the traffic passing through it respectively. $F(C)$ is the fixed power consumption of the network device, including processor, memory, cooling system, and so on. The power consumption of switching fabric used to manage the switching tables is also contained in $F(C)$ [23]. $Q(C)$ is the total power consumption of all enabled linecards plugged in the network device. The value is computed when all the ports on the linecards are disabled. $H(C,T)$ denotes the total power of all enabled ports in the network devices. The value of $H(C,T)$ depends on both port capacity configuration and traffic rate across them, and can be computed in the equation (2), where $\mathcal{I}$ denotes the space of all the possible port capacity configurations, such as 10/100/1000 Mbps. N_i denotes the number of enabled ports with capacity i, and B_i denotes the power consumed by each of the ports with a 100% utilization ratio. The power benchmarking results in [28] show that the power consumed by a port decreases as the traffic rate of the port decreases, and vice versa. As a result, the power consumed by the port with the capacity i and any utilization ratio is no more than B_i.

We take U_i in the equation (2) as an average difference value of power consumption so as to more accurately capture the real power consumption of the port with capacity *i*.

$$P(C,T) = F(C) + Q(C) + H(C,T) \quad (1)$$

$$H(C,T) = \sum_{i \in \mathcal{I}} N_i * (B_i - U_i) \quad (2)$$

It is not easy to compute the power consumption of a network device with the equations (1) and (2) in practice, because the network traffic changes frequently. Consequently, we simplify the model and discuss a more practical model as follows.

Simplified Model

As the power consumption of most network devices today is not proportional to their loads and the power consumption of an idle network device is close to that of the full load, the varying traffic rate does not affect significantly the power consumption of ports in current network devices. In fact, U_i in the equation (2) can be ignored when the port's capacity is large [28]. In addition, as current data center networks usually use plenty of commodity switches equipped with a single line-card to interconnect servers, we focus on this type of switches and simplify the power consumption model to the equation (3). *P(C)* denotes the total power consumption of a switch, which only relies on the configuration of the switch. $\mathcal{J}$ denotes the set of ports enabled on the switch and $A_j(C)$ is the power consumption of a switch port *j*. In the equation (3), the power consumption of line-cards is incorporated into the fixed power consumption *F(C)*. We observe that the total power consumption of a switch is independent of its traffic characteristics and thus we can easily compute the power consumption of a switch with the simplified model. Furthermore, the simplified model will introduce some errors and the power consumption of a switch computed with the equation (3) will be larger than the real power consumption of the switch. This is because the utilization ratio of a port will affect its power consumption [20] [28] and the port with a lower utilization ratio will consume less power. We take $A_j(C)$ as the approximate power consumption value of a switch port *j*, as the power consumption difference of a port under the varying traffic can be ignored when compared with the total power consumption of the switch. If all ports on a switch have the same power consumption *A*, we can further simplify the power consumption model into the equation (4), where *X* denotes the number of ports enabled on the switch.

$$P(C) = F(C) + \sum_{j \in \mathcal{J}} A_j(C) \quad (3)$$

$$P(C) = F(C) + A(C) * X \quad (4)$$

From the power consumption models above, we can get some inspiration on the potential strategies to save power consumption of network devices. First, the best way for power conservation is to put individual switches into the sleep mode or shut them down, because it can avoid the overall power consumption from a device, including the fixed power consumption (i.e. *F(C)* in Equations (1), (3) and (4)). However, the demands of high-performance and high-reliability of communications restrict the use of "device-level sleeping" method throughout the data center network. Another power conservation method is to put switch ports to "component-level sleeping" [7]. Obviously, the power-saving effect of the second method is below that of the first one, but it can be used as an effective supplementary means of "device-level sleeping" method to further reduce the power consumption. For example, for a typical commodity switch Cisco Catalyst 2960G-48TC-L which contains 1 line-card and 48 Gigabit Ethernet ports, the upper bound of power saving percentage can achieve 39% by disabling all ports of the switch if each port consumes 1W [32]. We combine both of the two methods in our algorithm, and call it *multi-granularity power saving strategy*, which can better exploit the power saving flexibility and more effectively

reduce the power consumption of network devices while meeting the network performance requirement.

2.2. Throughput-guaranteed Power-aware Routing Model

The objective of throughput-guaranteed power-aware routing is to compute the transmission paths based on a given traffic matrix, so that the total power consumption of switches involved in the routing paths is as little as possible while meeting predefined network performance thresholds. Throughout this paper we use network throughput as a key performance metric, since it is the most important metric for data-intensive computations. For the convenience of presentation, we call our throughput-guaranteed Power-aware Routing Problem PRP.

PRP: Assume there is a quadruplet of input parameters, (*G, T, K, RP*). *G* denotes the topology of a data center network, which consists of servers and switches along with their connections and link capacity. *T* is the network traffic matrix, which denotes the communication relationship between each pair of servers in the topology *G*. *K* and *RP* denote the predefined thresholds of total network throughput and network reliability respectively. The objective of PRP is to find a routing *R** for *T*, under the constraints of equations (5), (6) and (7). *R* denotes one of routing selections from a candidate set $\mathcal{R}^+$ which includes all possible routing paths for *T*, *L*(*R*) and *X*(*R*) denote the number of switches and ports involved in *R* respectively, *M(R)* denotes the total network throughput of all flows in *T* under *R*, and *Z(R)* denotes the minimum number of available paths for each flow in *T* under *R*. We assume the fixed power consumption *F* of each switch is the same and all switch ports in data center networks have the same capacity and power consumption *A*, as current advanced data center topologies, such as Fat-Tree and BCube, usually use homogeneous commodity switches to interconnect servers. We summarize the main notations used for our power-aware routing model and algorithm in the TABLE I.

$$R^* = \arg\min_R (F * L(R) + A * X(R)) \tag{5}$$

$$M(R) \geq K, R \in \mathcal{R}^+ \tag{6}$$

$$Z(R) \geq RP, R \in \mathcal{R}^+ \tag{7}$$

The purpose of our model design is to find the optimal power-saving routing, in which the total power consumption of switches involved is minimal while satisfying the network throughput and reliability thresholds. In this paper we focus on bandwidth-hungry batch-processing tasks in data centers, such as MapReduce [1] and GFS [2]. For this kind of application, total network throughput is a better metric to reflect the computation progress, instead of the constraints of single flows. Therefore, we use the threshold *K* to restrict the lower bound of the total throughput of all flows in the equation (6).

Also, it is necessary to ensure any network flow identified in the traffic matrix will not be interrupted when using the power-aware routing. As PRP is an NP-hard problem, it is difficult to find the optimal solution of the problem in polynomial time, especially when the scale of the data center network is large. We have proved NP-hardness of the problem by reducing the 0-1 Knapsack problem [10] to it. The details of NP-hardness proof can be found in the appendix.

It is worth noting that the model we use is different from some previous similar work. In [7] and [41], each network flow has a fixed traffic rate, which is taken as the input of the power minimization problem. In our PRP model, the transfer rate of each flow in the traffic matrix is impacted by the bandwidth competition among different flows, as the network is currently the bottleneck for distributed computation in data centers. Moreover, TCP traffic is dominant in current data center networks. TCP uses the sliding window scheme and AIMD (Additive Increase Multiplicative Decrease) algorithm to control the transfer rate of flows. The work in [11] reveals that the rate of competing TCP flows mainly relies on their RTT (Round-Trip Time). In high-bandwidth and low-latency data center networks, TCP flows have a similar RTT value and share fairly the available

bandwidth of the bottleneck link. Therefore, TCP follows the Max-Min fairness model [12] to assign the transfer rate for each flow in DCNs.

The throughput-guaranteed power-aware routing model is applied in data center networks, which have their own characteristics compared with the Internet. First, current advanced DCN architectures use an excessive number of network resources to provide routing services and enjoy abundant routing paths between each pair of servers. Also, existing DCN architecture designs usually employ symmetric topology structures [52] and homogeneous network devices. Second, data center networks have more flexible traffic control and management schemes, such as OpenFlow [7], which make the design and implementation of power-aware routing more tractable. Third, there are some typical traffic patterns in data centers, such as One-to-One, One-to-Many and All-to-All traffic patterns.

TABLE I
Summary of main notations

Notation	Description
G	topology of a data center network
T	network traffic matrix
K	network throughput threshold
$\mathcal{R}^+$	set of all possible routing paths
R	a possible routing selection from R^+
R^*	optimal power-aware routing
F	Fixed power consumption of switches
A	power consumption of each switch port
$L(R)$, $X(R)$	number of switches and ports involved in R
$M(R)$	total network throughput of all flows under R
PR	network performance threshold percentage
RP	reliability requirement parameter
Y_i	rate of flow i
C	capacity of a link

2.3. Related Work

In the research area of green networking, some survey papers [22], [36] have summarized many novel network power saving technologies appearing in recent years. In this subsection, we only discuss typical data center network architectures and representative power conservation schemes used in Internet and data centers.

Data Center Network

Due to the well-known problem of the current practice of tree topology, recently there are a bunch of proposals on new topologies for data centers. These topologies can be divided into two categories. One is switch-centric topology, i.e., putting interconnection and routing intelligence on switches, such as Fat-Tree [4], Portland [13], and VL2 [14]. Contrarily, the other category is server-centric, namely, servers with multiple NIC ports also participate in interconnection and routing. BCube [5], DCell [6] and FiConn [15], all fall into the latter category.

Green Internet

The problem of saving overall power consumption in the Internet was firstly proposed by Gupta et al. [16]. Later, they applied low-power modes and novel detection methods of inactive period into local area network for power saving [17]. In the follow-on work, lots of efforts focused on device power management [18], [19], realizing two modes, i.e. sleeping and rate-adaptation, on network devices to reduce the power consumption during network idle or low-load time. Chabarek et al. [20] designed and configured network and protocol from the power saving perspective, and optimized the power consumption of network devices with mixed integer optimization techniques.

There are also some recent studies on power-aware traffic engineering and routing in the Internet. Zhang et al. [37] proposed an intra-domain green traffic engineering scheme, which minimized the power consumption of network links while satisfying the network performance restriction. Similarly, Vasic et al. [38] added the power conservation objective into the traditional traffic engineering and dynamically adjusted load distribution on multiple available paths to save network power consumption in the Internet. Moreover, Kim et al. [44] re-formulated the network energy consumption minimization problem and presented an ant colony-based routing scheme to improve energy efficiency of the Internet. Avallone et al. [46] proposed an energy-efficient online routing scheme, which used the fewest number of network resources to transmit flows while meeting several additive network performance constraints. Cianfrani et al. [39] [40] proposed power-aware OSPF routing protocols, which used as few links as possible to provide routing services by modifying Dijkstra's algorithm and sharing the shortest path trees of elected routers under light loads. Cuomo et al. [45] studied the algebraic connectivity and link betweenness of topology graphs and proposed a traffic-unaware energy-saving topology control scheme. In [31], Fisher et al. presented a novel method to reduce the power consumption in Internet backbone networks by shutting off individual physical cables belonging to bundled links. In [43], Vasic et al. leveraged both energy-critical path pre-computation and on-line traffic engineering to achieve more effective power conservation as well as lower routing calculation complexity. Furthermore, many recent works were devoted to improving energy efficiency of optical networks, such as [47-50].

As all the above designs target for the Internet, they need to be compatible with current Internet protocols, such as routing protocols and traffic control and management protocols. In contrast, we consider saving power in data center networks and can thus fully leverage the characteristic of "richly-connected" topologies and flexible traffic management schemes to better achieve the power-aware routing in current DCNs.

Green Data Center

There were also more and more concerns with power saving in data centers. Plenty of researches and technologies have been proposed recently, which can be divided into four categories as follows.

(1) Power-efficient hardware

Magklis et al. proposed a profile-driven approach to adjust processors' voltage and frequency dynamically to save significant power with little performance loss [21]. In [23], three schemes were proposed to reduce the power consumed by switches, including the prediction of sleep time, set-up of power saving mode and deployment of low-power backup devices. The "hardware-level" technologies above aim to reduce the power consumption of individual network devices from the hardware design and implementation perspective. In contrast, our work focuses on the "network-level" power conservation scheme, and its objective is to reduce the total power consumption of the whole data center network from the routing perspective. These "hardware-level" technologies above are effective supplements to our "network-level" scheme, and the two types of power conservation techniques can work cooperatively.

(2) Power-aware routing in DCNs

The use of power-aware routing in DCNs is still in its infancy, but some of researches have been on the way. Heller et al. proposed a novel scheme named ElasticTree [7], which could save network power effectively by computing a group of active network elements according to network loads. They built a prototype system based on the OpenFlow switches to evaluate the power conservation effect, network performance and fault-tolerance when applying the ElasticTree to DCNs. In contrast, our scheme is based on a different traffic rate model as presented earlier. The transfer rate of each flow depends on network competition with other flows, instead of given by the network traffic demand in advance.

In a recent work, we proposed a power-aware routing scheme in data center networks [30]. In this paper, we make a significant technical extension of the work in [30], and include more detailed designs of models and algorithms. We also conduct a large number of new power conservation and performance studies to evaluate our routing algorithm.

(3) Scheduling and virtualization

Nedevschi et al. presented a proxy model to handle all kinds of idle-time behaviors in networks and evaluated the relationship among power saving benefit, proxy complexity and available functions [25]. Srikantaiah et al. optimized power consumption by reasonably scheduling applications and studied the tradeoff between the power consumption and service performance of devices [26]. Nguyen et al. [51] studied how to reduce greenhouse gas emissions from ICT and leveraged virtualization technologies to process user service applications in greener data centers. Moreover, optimal virtual machine migration and placement scheme was studied and used to build data center architectures for power saving [33].

(4) Improving PUE

Power Usage Effectiveness (PUE) is a key metric to measure the power efficiency of data centers. Many advanced and smart cooling technologies were proposed to improve PUE, such as [24]. Recent experiments in [27] show that deploying data centers in Frigid Zone is also a feasible method to reduce cooling costs.

3. Throughput-guaranteed Power-aware Routing Algorithm

In this section, we propose a throughput-guaranteed power-aware routing algorithm, which can be applied in an arbitrary DCN topology. The objective of our routing algorithm is to compute the transmission paths for all the flows identified in the traffic matrix, so that the total power consumption of switches involved in the routing paths is as little as possible. We use a multi-granularity power saving strategy to reduce power consumption from both the device-level and the component-level perspectives. As computing the optimal solution of PRP is NP-hard, which has been shown in Section 2.2, our routing algorithm tries to find an effective and more practical power-aware routing.

The basic idea of our throughput-guaranteed power-aware routing algorithm is as follows: First, taking all the switches and links in a given topology into consideration, we compute the routes for the traffic flows and the corresponding network throughput which are called *basic routing* and *basic throughput* respectively. Second, we gradually remove the switches from those involved in the basic routing, based on the specific elimination order in accordance with the workload of switches, until the network throughput decreases to the throughput threshold the system can tolerate. We get an updated topology after the iteration process of switch elimination. Third, we remove as many links connected to the active switches as possible from the updated topology above based on a link elimination strategy while ensuring the network throughput to meet the throughput threshold. Finally, we obtain the power-aware routing paths for the traffic flows and adapt the topology derived from the previous steps to meet a predefined reliability requirement. After these steps, we can power off the switches and links not involved in the final topology or put them into the sleep mode for power conservation. In the routing algorithm, we use connection-unsplittable routing [11] to avoid packet disorder, i.e., the packets from one flow takes only one path.

Note our proposed algorithm is pruning-based, and it iteratively eliminates switches and links from the original topology. An alternative routing algorithm could be computing and assigning the power-aware routing paths for flows one by one, that is, adding in flows incrementally. For example, among multiple available paths for a flow, the path containing the fewest number of idle network nodes and links can be chosen preferentially, so as to reduce the number of switches and links used. The greedy “incremental” routing algorithm may severely hurt the network performance while pursuing the goal of power conservation. We take an example to illustrate this problem. In Fig. 1, there are four switches and four links, and the capacity of each link is 1Gbps. There are three flows: A->B, B->C, A->C in the network. The “incremental” routing algorithm will sequentially assign the power-aware routing paths for them. The former two flows take the paths A->B and B->C respectively, each of which only contains one idle link. The two flows exclusively enjoy the 1Gbps bandwidth of link A-B and B-C respectively and thus the total network throughput is 2Gbps. Then, the path A->B->C will be used to transfer the

third flow A->C by the “incremental” algorithm, as all the switches and links on the path have been used by the former flows, and transferring the new flow does not need any additional idle switch or link. However, at the same time the flow A->C will compete with the former two flows for the bandwidth of link A-B and B-C, and the total network throughput will unexpectedly decrease from 2Gbps to 1.5Gbps according to the Max-Min fairness model [11] [12]. Therefore, the “incremental” routing algorithm should contain the network performance constraint when choosing routing path for each flow. In our pruning algorithm, we compute the basic routing and basic throughput at the beginning, and use the performance thresholds to restrict the extent of network performance degradation during the switch elimination and link elimination processes. Our pruning algorithm is an important alternative to the existing “incremental” routing algorithm, and both of them can save network power under the restriction of network performance. We will show the comparison results of power conservation effectiveness of the two types of algorithms in Section 4.6.

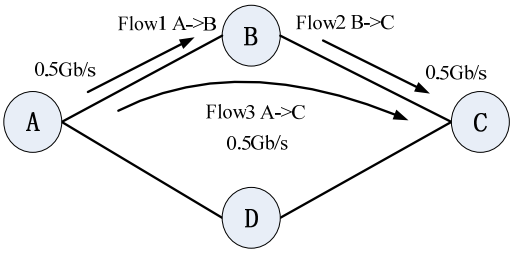


Fig. 1. An example of path assignment with the “incremental” routing algorithm.

Our routing algorithm consists of five modules: Route Generation (RG), Throughput Computation (TC), Switch Elimination (SE), Link Elimination (LE) and Reliability Adaptation (RA). The relationship among the five modules is shown in Fig. 2. We execute the SE module and the LE module sequentially to implement a multi-granularity power saving strategy. The SE module is executed repeatedly prior to the LE module, until eliminating one more switch from the topology will violate the network performance restriction. Then the LE module is executed repeatedly following it. When the network throughput decreases to the throughput threshold given by the input of the algorithm, we finish the link elimination phase. Finally, the RA module updates the topology by adding some of the removed switches and links back to the topology to meet the reliability requirement, and outputs the updated topology with transmission paths that are power-aware and reliable. The module execution order is marked with numbers in Fig. 2.

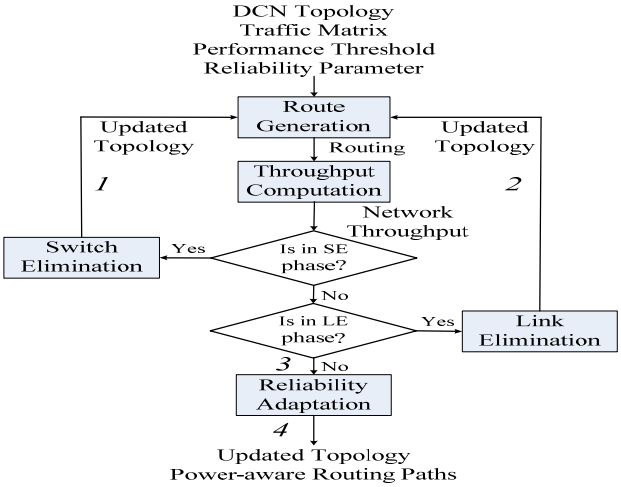


Fig. 2. Relationship among five modules of our algorithm.

Throughput-guaranteed Power-aware Routing Algorithm: PRA(G_0, T, PR, RP)

Inputs:
G_0: DCN topology
T: traffic matrix
PR: performance threshold percentage
RP: reliability requirement parameter
Outputs:
$R1$: power-aware routing paths
$G1$: updated topology
Begin
1 set $G := G_0$;
2 //Route Generation
3 set R := RG(G, T);
4 //Throughput Computation
5 set $Tht1$:= TC(G, T, R);
6 // Switch Elimination
7 **do**
8 **begin**
9 set $G1 := G$;
10 set $R1 := R$;
11 set G := SE(G, T, R);
12 set R := RG(G, T);
13 set $Tht2$:= TC(G, T, R);
14 set $P := Tht2 / Tht1$;
15 **end while**($P >= PR$)
16 // Link Elimination
17 $G := G1$;
18 $R := R1$;
19 **do**
20 **begin**
21 set $G1 := G$;
22 set $R1 := R$;
23 set G := LE(G, T, R);
24 set R := RG(G, T);
25 set $Tht2$:= TC(G, T, R);
26 set $P := Tht2 / Tht1$;
27 **end while**($P >= PR$)
28 $G1$:= RA($G1$, T, $R1$, RP)
29 **return** ($R1$, $G1$);
End.

Fig. 3. Throughput-guaranteed power-aware routing algorithm.

Fig. 3 shows the pseudocode of our throughput-guaranteed Power-aware Routing Algorithm (PRA). The input of the algorithm is the quadruplet (G_0, T, PR, RP). G_0 denotes the data center network topology, T denotes the traffic matrix. PR is the *performance threshold percentage*, which is defined as the ratio of the network throughput the data center operator can tolerate after the switch and link elimination phases, to the basic throughput using all switches. In the algorithm, we translate the parameter K in the throughput-guaranteed power-aware routing model into PR for explicitly measuring the performance degradation resulting from the power-aware routing. The parameter K is equal to PR multiplied by the basic throughput. RP is used to set the reliability requirement for the power-aware routing. The outputs of the algorithm are the power-aware routing paths $R1$ for T and the updated topology $G1$. In Fig. 3, we compute the basic routing paths R and the basic throughput in topology G_0 shown on lines 3 and 5 respectively. From line 6 to line 15, we perform switch eliminating operation.

We iteratively execute three modules: SE, RG and TC, until P is less than PR, where P is the ratio of the throughput *Tht2* in G to the basic throughput *Tht1* in G_0. Afterwards, from line 19 to line 27, we remove as many links as possible from the updated topology for saving further power under the performance constraint. Finally, we execute the RA module to meet the reliability requirement based on the *RP* value setting in line 28.

In the following five subsections, we will present the five modules in detail, including each module's function, design and implementation mechanism, and an example taken for easily understanding the ideas proposed.

3.1. Route Generation

The role of the route generation module is to select the route path for each flow in the traffic matrix so that the network throughput is as high as possible. Each selected path can be computed with the inputs: network topology and traffic matrix, and the output of RG module is the routing paths for all flows belonging to the traffic matrix in the topology.

We describe the detailed path selection method for each flow below. Assume there are a certain number of available paths for a flow and the set of available paths, denoted by *PathSet*, is taken as the input of the path selection method. For a certain flow, the ideal solution is to enumerate all possible paths. However, it may not be practical for large-scale networks. Hence, there is a tradeoff between the computational complexity and the efficiency of the results. It is straightforward to find multiple available paths for a flow in DCNs by leveraging the topological characteristic. For example, a *k-ary* Fat-Tree topology contains $(k/2)^2$ available paths traversing different core switches respectively between any two inter-pod servers, and the maximum number of available paths between any two intra-pod servers is $k/2$ [4]. In a k-level BCube topology, each server uses $k+1$ NIC ports to connect to different level switches and there are $k+1$ parallel paths between any pair of servers [5]. The path selection method uses four rules in turn to select the best one path from *PathSet* for the flow. In Rule1, we first select the paths with the fewest overlapping flows over the bottleneck link in the paths. The basic reasoning behind Rule1 is to exploit multiple paths sufficiently and achieve as high network throughput as possible. If there are multiple such paths, we use Rule2 to further filter them. Rule2 selects the paths with the fewest hops from *NewPathSet*, in order to have a lower propagation delay of network flows and use as few network resources as possible to achieve the same network throughput. If there is still more than one path, Rule3 works and chooses the path with the maximum number of flows, which is calculated by summing up the number of flows on every link of the path. The intuition behind Rule3 is trying to aggregate network flows, so that as few network resources as possible are used to transmit these flows. Finally, Rule4 chooses the first path from multiple available paths output by Rule3. We finish the path selection process when the selected path output by any of the four rules is unique, and obtain the best path for the flow. The computational complexity of RG module is $O(FN*p)$, where FN denotes the number of network flows and p denotes the number of available paths between each pair of servers in the topology.

We take a partial 4-ary Fat-Tree topology [4] as an example to explain the details of the path selection method in Fig. 4 and TABLE II. We select the path for each flow in TABLE II in turn. The first flow, of which the source and destination servers are H1 and H4 respectively, has two available paths, one across switch S5, and the other across switch S6. Obviously both paths conform to Rule1, Rule2 and Rule3, so the first path is chosen according to Rule 4. The path is marked with small red squares in Fig. 4 and shown in TABLE II. The second flow "H2->H7" has four available paths, traversing S1, S2, S3 and S4 respectively. The latter two paths are chosen by Rule1, as the first flow has already used link S5-S9. The two paths also conform to Rule2 and Rule3, so the path across the switch S3 is chosen by Rule4, marked with yellow triangle in Fig. 4. All four available paths of the third flow conform to Rule1 and Rule2. We find that the total number of flows on all links contained in the path across switch S3 is five, which is maximal among the four available paths for the third flow, and thus we choose the path according to Rule3, marked with blue circles in Fig. 4.

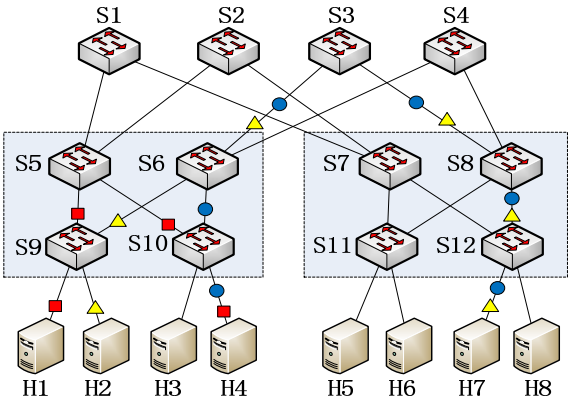


Fig. 4. A path selection example in a partial 4-ary Fat-Tree topology.

TABLE II
Network flows in a path selection example

No.	Flows	Marks	Path Selection
1	H1–>H4	■	H1–>S9–>S5–>S10–>H4
2	H2–>H7	△	H2–>S9–>S6–>S3–>S8–>S12–>H7
3	H4–>H7	●	H4–>S10–>S6–>S3–>S8–>S12–>H7

3.2. Throughput Computation

The module of throughput computation is used to calculate the total network throughput in a given DCN topology. We use a well-known model, named Max-Min Fairness model [11], [12], to allocate network resources to all flows in a DCN. When given the network topology, traffic matrix and routing path for each flow, we can compute the rate of each flow in the traffic matrix with the model, and sum them up to acquire the total network throughput. The throughput computation algorithm is shown in Fig. 5, and the computational complexity of TC(.) is $O(LN^2)$, where LN denotes the number of links in the topology.

Throughput Computation: TC(*G, T, R*)

Input: *G*: DCN topology, *T*: traffic matrix, *R*: routing paths
Output: *Tht*: the total throughput of all flows in *T*
Notations:
AB_j: available bandwidth of link *j*
FN_j: the number of flows unallocated rate on link *j*
FS_j: the set of flows unallocated rate on link *j*
Y_i: the rate of the flow *i*
FS: the set of flows unallocated rate in *T*
*FS*0: the set of all flows in *T*
*LS*0: the set of all links in *G*
Begin
set $FS := FS0$, set $LS := LS0$;
while(*FS* is non-empty)
begin
(1) Find the bottleneck link $L \in LS$, such that
$AB_L / FN_L \le AB_j / FN_j, \forall j \in LS, AB_j, AB_L, FN_j, FN_L > 0$;
(2) set $LS := LS - \{L\}$, set $FS := FS - FS_L$;
(3) set $Y_i := AB_L / FN_L, i \in FS_L$;

(4) set $AB_L = 0, FN_L = 0, FS_L = \varnothing$;

(5) Update $AB_j, FN_j, FS_j, j \in LS$;

end

$Tht = \sum_{i \in FS0} Y_i$;

return *Tht* ;

End.

Fig. 5. Throughput computation algorithm.

We also take the example in Fig. 4 and assume that the capacity of all links is 1Gbps. We find that five links: S10-H4, S3-S6, S3-S8, S8-S12, S12-H7 can be seen as the bottleneck links, so we select one of them arbitrarily, supposing link S12-H7. As such, the rate of two flows: H2->H7 and H4->H7 calculated by the throughput computation algorithm are both 0.5Gbps in Fig. 5. After updating the variables AB_j, FN_j and FS_j of all links, the next bottleneck link is S10-H4, so the rate of flow H1->H4 is 0.5Gbps. Therefore, the total network throughput calculated by the TC module is 1.5Gbps.

3.3. Switch Elimination

The switch elimination module is responsible for selecting the switches which can be eliminated from the routing paths generated for the traffic matrix. First, we compute the traffic ST_j carried by each active switch j in the topology G, which denotes the total throughput of flows traversing the switch j. Then, we use a greedy algorithm to select the switch *sw* and eliminate it from the topology according to the equation (8), where $\mathcal{F}(j)$ is the set of flows traversing the switch j, Y_i is the rate of flow i, and $\mathcal{S}$ is the set of all active switches. In other words, the algorithm indicates that the active switch carrying the lightest traffic should be eliminated first. If there are multiple switches meeting the elimination condition above, we prefer to remove the switch with the maximal number of active links connected to it. The strategy intends to have more opportunities to disable the ports of its neighboring switches. Moreover, to accelerate the computation process, we usually eliminate more than one switch from the topology G per round in practice. Also, the switches eliminated from G cannot be the critical ones that may lead to network disconnection. The computational complexity of SE module is O(*SN*), where *SN* denotes the number of switches in the topology.

$$sw = \arg\min_{j \in \mathcal{S}}(ST_j) = \arg\min_{j \in \mathcal{S}}(\sum_{i \in \mathcal{F}(j)} Y_i) \tag{8}$$

3.4. Link Elimination

We use the link elimination module to remove unused or low-utilized links from the topology G updated by the SE module, after finishing the switch elimination process. The link elimination module is used as an effective supplementary means to further reduce the power consumption by network components after the switch module completes its jobs.

We use a greedy removing algorithm to select the links to be eliminated from the topology G. First, we compute the utilization ratios for all active links connected to the active switches in the topology G. The utilization ratio of a link j can be calculated based on the total throughput of flows traversing the link and the capacity of the link in the equation (9), where LU_j is the utilization ratio of link j, $\mathcal{F}'(j)$ is the set of flows on the link j, R_i is the rate of flow i, and C is the capacity of the link j.

$$LU_j = (\sum_{i \in \mathcal{F}'(j)} R_i) / C \quad (9)$$

Then, we select the link with the lowest utilization ratio, and eliminate it from the topology *G*. Finally, we can disable the ports incident to the link for saving power. Like the SE module, the LE module cannot remove the link in critical paths either in order to not interrupt the flows identified in the traffic matrix. The computational complexity of LE module is O(*LN*), where *LN* denotes the number of links in the topology.

3.5. Reliability Adaptation

In order to minimize the power usage, SE and LE modules may aggressively reduce the redundancy of the "richly-connected" data center network topologies, which compromises the transmission robustness. To address this issue, we introduce an additional reliability adaptation module into our routing framework to maximally reduce the power consumption while meeting the transmission reliability requirement. Generally, there is a tradeoff between the reduction of the number of switches/links and the need of maintaining enough redundancy level of the topology for transmission robustness. In the RA module, we set the reliability parameter *RP* to denote the minimum number of available routing paths needed for each flow in the traffic matrix. The value *RP* can be configured based on the reliability requirement of the DCN, and the switches and links on the backup paths cannot be eliminated from the topology.

After finding at least one available path for each flow through the SE and LE modules, if there is a need to introduce additional redundancy to meet the reliability requirement, the RA module is invoked to find and preserve additional *RP*-1 paths for each flow based on the *RP* value configured for the DCN. Assume there are *n* available paths in the initial topology G_0, for a flow *i*. The RA module needs to select additional *RP*-1 backup paths from *n*-1 available paths for each flow in turn, besides the power-aware routing path set by the SE and LE modules already. If the newly added paths need to go through some switches and links already removed by SE and LE from the original network topologies, the RA module will add the switches and links back to the updated topology. When selecting each additional backup path for a flow *i*, the RA module gives the preference to the path P_i based on the following equation (10), where $\mathcal{PS}_i$ denotes the set of the available paths which have not been selected as the backup or power-aware routing paths yet for the flow *i* in the initial topology G_0. *N(j)* denotes the number of switches which have been eliminated by the SE module from the path *j*. The intuition behind the backup path selection strategy is that we want to use as few additional switches as possible to meet the reliability requirement. The computational complexity of RA module is O(*FN***p*), where *FN* denotes the number of network flows and *p* denotes the number of available paths between each pair of servers in the topology.

$$P_i = \arg\min_{j \in \mathcal{PS}_i} N(j) \quad (10)$$

The quality of backup paths depends on the number of nodes shared among paths selected. Ideally, the number of nodes shared between the primary path and back paths as well as shared among several backup paths should be as low as possible. In data center networks, however, the quality of backup paths selected by the RA module also depends on the network topologies. In a *k*-level BCube topology, there are *k*+1 node-disjoint paths between a source-destination server pair [5], and thus the RA module can ensure all selected backup paths are node-disjoint. While in the Fat-Tree topology, as all of available backup paths between any source-destination server pair will pass through the same edge switches [4] and thus no node-disjoint path exists. In fact, after finishing the switch elimination and link elimination processes, the RA module can always find the backup routing paths of which the quality is not worse than that of available paths in the original topology to effectively improve the network reliability.

It is worth pointing out that the reliability adaptation in our throughput-guaranteed power-aware routing algorithm is an optional function. Data center operators can decide whether to use it according to the quality of network devices and network reliability requirements in current data centers. Therefore, we use an independent module to implement the reliability adaptation function so as to more flexibly enable and disable it by configuring related parameters. Actually, when executing the RA module in practice, we can combine it with the SE and LE modules and take the reliability constraints into account in the switch and link elimination phases. In the paper, we logically separate the RA module from the SE and LE modules for more explicitly presenting its objective and function.

4. Evaluation

In this section, we evaluate our throughput-guaranteed power-aware routing algorithm through simulations using BCube and Fat-Tree topologies respectively. Both types of topologies are commonly used for data center networks. We evaluate the effectiveness of our algorithm based on two kinds of power-saving granularities: the "device-level" granularity which only configures the individual switches to the low-power mode, and the "component-level" granularity which puts both the switches and links into the low-power mode. The evaluation results for the two granularity levels are shown in Sections 4.2 and 4.3 respectively. We then evaluate the algorithm using several typical traffic patterns, including One-to-One, One-to-Many and All-to-All in data centers, and the results are shown in the Section 4.4. Afterward, we study the tradeoff between conserving more power and meeting the reliability requirement in Section 4.5. Finally, we evaluate the power saving effectiveness of our algorithm by comparing with the ElasticTree scheme [7] as well as the optimal solution of throughput-guaranteed power-aware routing problem in Section 4.6.

4.1. Simulation Setup

The BCube and Fat-Tree topologies used in the simulations are set to different sizes. For a recursive structural topology BCube(*BN*, *BL*), *BN* denotes the number of ports in a switch, and *BL* denotes the number of levels counted from the level_0. A level_*i* BCube(*BN*, *i*) contains *BN* level_*i*-1 BCube(*BN*, *i-1*) and additional BN^i switches[5]. We set different values for *BN* and *BL* to vary the topology sizes. Similarly, we can vary the number of ports in each switch of the Fat-Tree(*FP*) topology, denoted as *FP*. Furthermore, we analyze the real traffic data inside data centers in [14], which provides the time duration of different loads on an average server during a day. We observe that each server has at most one concurrent flow during the 1/3 time of a day and thus we consider network power conservation and evaluate our algorithm during this period. In our simulations, we take the maximum of the total number of flows on all servers as 100% network load and vary the percentage of the number of flows in the topology to simulate different network loads. The traffic matrix is randomly generated between servers in Sections 4.2, 4.3, 4.5 and 4.6, and three typical traffic patterns usually found in real data centers are used to evaluate the algorithm in Section 4.4. The One-to-One traffic pattern is often used to copy large files from one server to the other; The One-to-Many traffic pattern is frequently used for data chunk replication from one server to several servers to ensure higher reliability [5] and often seen in Google File System [2], CloudStore [3] and other scenarios; MapReduce [1] jobs running in data centers will produce the All-to-All traffic pattern. Moreover, we set the reliability parameter *RP* as 1 in Sections 4.2, 4.3, 4.4 and 4.6, and vary the parameter value to evaluate the effect on power conservation under different reliability requirements in Section 4.5.

To evaluate the effectiveness of power conservation, we use the power saving percentage as a metric, which is defined as the network power saved by our algorithm over the total power consumption of the network without using any power conservation strategy. We take the power consumption data of a general access-level switch, Cisco Catalyst 2960G series as an example to evaluate the power conservation effect of our algorithm. For DCN

topologies with different sizes, we choose the switches equipped with different numbers of ports. We use the simplified network power consumption model presented in Section 2.1 and the real power consumption data of the switches in TABLE III [32], which are measured under full loads, to compute the total power consumption of DCNs before and after using our algorithm. Generally, the switches which are put into the low-power mode or shut down consume very little power, and disabling one port of the switch saves about 1W power [32]. We set the performance threshold percentage as 95% in the following simulations, except when studying the relationship between the power consumption and the performance threshold percentage.

TABLE III
Cisco 2960G switch power consumption data

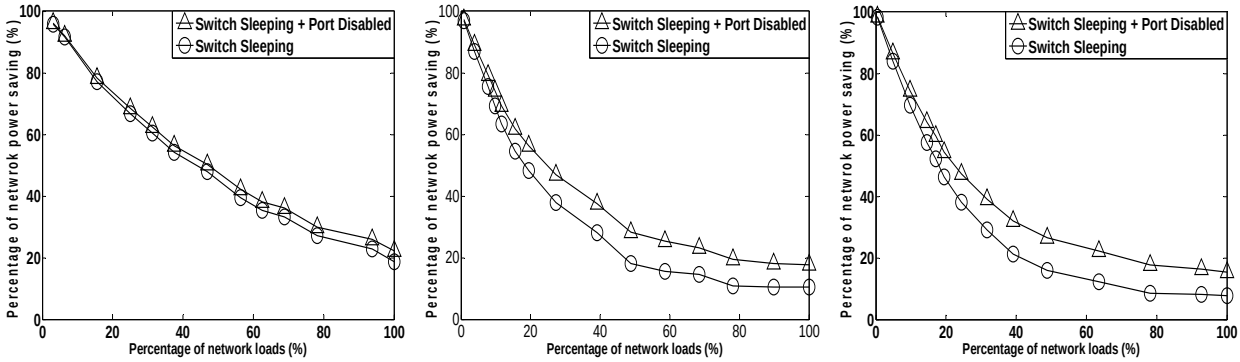
Switch Type	Power (W)	Applied to Topologies
WS-C2960G-48TC-L	123	Fat-Tree(48)
WS-C2960G-24TC-L	72	Fat-Tree(12), Fat-Tree(24)
WS-C2960G-8TC-L	36	Fat-Tree(4), BCube(4,2), BCube(8,2) BCube(8,3)

4.2. Switch Sleeping

In this subsection, we evaluate the power saving percentage of our algorithm by putting a subset of switches into the low-power mode or shutting them down in data center networks with BCube or Fat-Tree topologies.
We evaluate the effect on power saving due to turning off switches with the increase of network loads for different topology sizes of BCube in Fig. 6. The curves marked with “Switch Sleeping” denote the power saving effect based on the “device-level” granularity, i.e. only using the switch sleeping strategy (the same below). We can see that our power-aware routing algorithm significantly reduces the power consumed by switches at low network loads. For example, in BCube(4,2), when the percentage of network loads is less than 80%, more than 30% power can be saved, and the power saving percentage increases to 60% if the network load is less than 30%. As expected, the power-saving percentage reduces as the network load increases. Fig. 7 shows the simulation results with the Fat-Tree topology. Our algorithm can also significantly reduce the power consumption of switches under the low load. The network power consumption in Fat-Tree(FP=48) can be reduced by more than 26% when the network load is less than 95%, and the saving increases to 50% when the network load is smaller than 10% in Fig. 7(c).

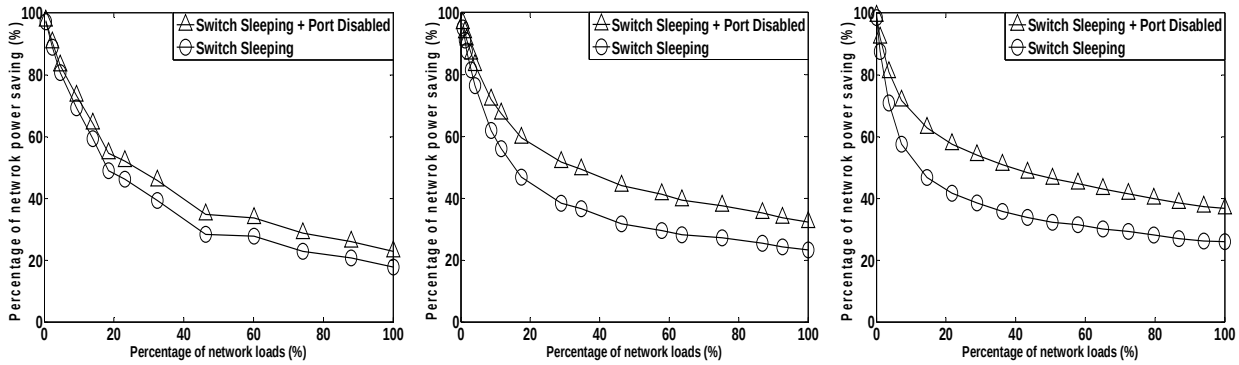
In our simulations, we generate network flows by randomly selecting their source and destination servers, and incrementally add them into the network to compute their routing paths in a random order. We also study the impact on the network throughput and power saving percentage by conducting simulations when adding flows into the network with different orders. For example, we can add flows based on the sequence number of their source or destination servers. The results reveal that different orders of adding flows have little effect on the both network throughput and effectiveness of power conservation in the BCube and Fat-Tree topologies. Moreover, in our evaluations, we execute the SE module to eliminate only one switch from the topology in each round. The SE module usually removes more than one switch per round in practice to accelerate the computation process. We conduct simulations to investigate how to decide the number of switches removed per round so as to more effectively accelerate the switch elimination process. The simulation results indicate that the value should be set according to the number of switches and network flows in the DCN. Under the low network load, the SE module has better execution efficiency when eliminating about 5-10% of the total number of switches per round. The percentage should be adjusted to a lower value as the network load grows. The simulation results above are not shown in the paper due to the page limit of the paper.

Fig. 8 shows the relationship between the power conservation and the performance threshold percentage in the BCube and Fat-Tree topologies. The “Switch” in the legends of Fig. 8 denotes the “device-level” granularity presented earlier. In Fig. 8(a), we vary the percentage of the performance threshold for a given BCube topology,



(a) BCube(4,2) (b) BCube(8,2) (c) BCube(8,3)

Fig. 6. Network power conservation percentage based on "device-level" and "component-level" granularities under different loads in BCube topologies.



(a) FP=12 (b) FP=24 (c) FP=48

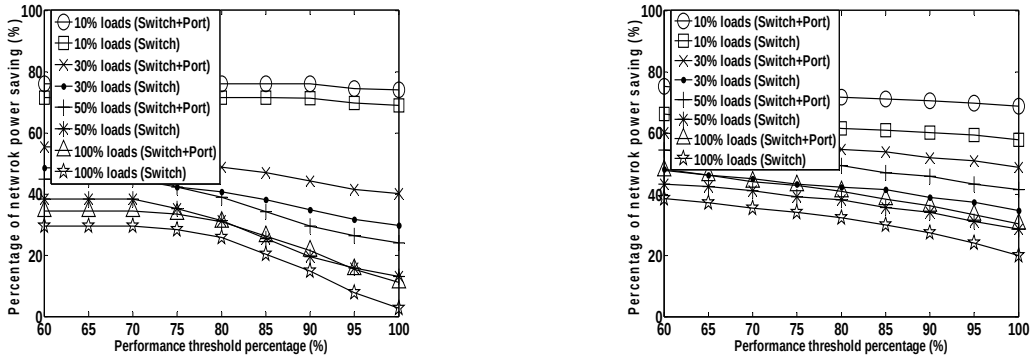
Fig. 7. Network power conservation percentage based on "device-level" and "component-level" granularities under different loads in Fat-Tree topologies.

i.e., BCube(8,3). Not surprisingly, as the performance threshold percentage increases, the power-conservation level decreases, because a higher performance requirement reduces the opportunities of removing switches from the topology. Hence, there is a tradeoff to consider according to the data center need and operational states. Actually a significant amount of power consumption can be reduced even without any performance sacrifice. This can be observed from the values corresponding to the performance threshold percentage 100% while the network has a low load. In Fig. 8(b), we conduct the simulation with the *FP* of the Fat-Tree topology set to 24. Similar to BCube, the effect of power saving decreases as the performance threshold percentage increases.

4.3. Port Disabling

In our power-aware routing algorithm, disabling ports (eliminating links) of active switches can be taken as a supplementary means to further save power after putting off parts of the network switches. Therefore, we want to calculate how much power can be saved by our algorithm after finishing the switch sleeping process.

The curves marked with "Switch Sleeping + Port Disabled" in Fig. 6 and Fig. 7 show the power conservation percentage based on the "component-level" granularity. One important observation is that we can further reduce a significant amount of power by disabling ports of active switches after putting a selected set of switches into sleep. For example, we can increase the power-saving percentage from 16% to 27% by disabling ports after



(a) BCube(8,3) (b) Fat-Tree(24)

Fig. 8. Network power conservation percentage based on "device-level" and "component-level" granularities under different performance threshold percentages in BCube(8,3) and Fat-Tree(24) topologies.

sleeping switches under 50% of network loads shown in Fig. 6(c). Similar results can be found in the Fat-Tree topology in Fig. 7. The network power of Fat-Tree(FP=24) can be saved about 30% and 41% when using two power-saving granularity strategies respectively under 60% of network loads in Fig. 7(b). The results from the simulations above show a positive effectiveness when using the port disabling method, and thus it can be taken an important supplement to the switch sleeping method to save more power.

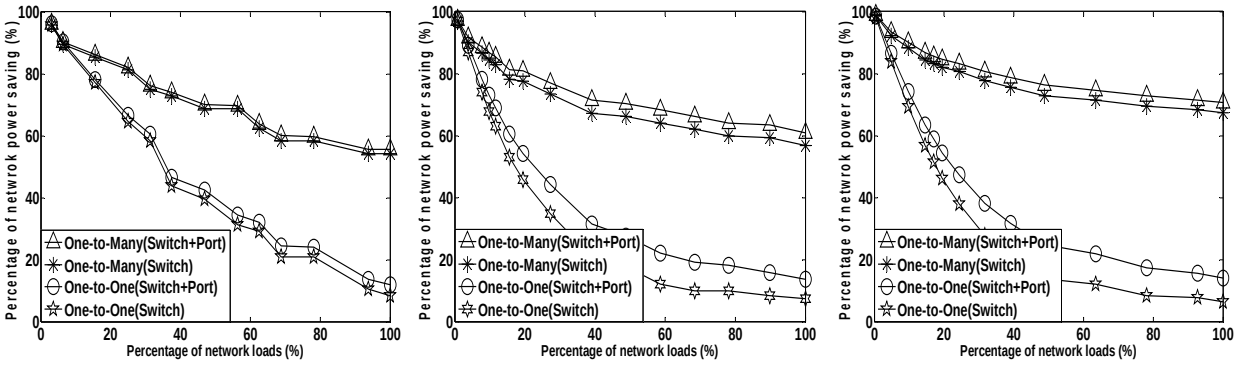
In Fig. 8, the curves marked with "Switch+Port" denote the "component-level" granularity method presented before. As expected, it can save more power when under the looser performance constraints or low network loads. For example, in terms of power-saving effect, it can save about 30% and 46% of total network power consumption in BCube(8,3) and Fat-Tree(24) respectively under the 50% of network loads and 90% of performance threshold in Fig. 8. And the two power-saving ratios can reach 45% and 55% respectively when the performance threshold percentage decreases to 60% under the same network loads. Moreover, Fig. 8 also shows the port disabling method can save substantial power after switch sleeping under different performance threshold restrictions.

4.4. Typical Traffic Patterns

We have shown that our power-aware routing algorithm can achieve a considerable power saving effect under the random traffic pattern from the studies above. In this subsection, we evaluate our algorithm under three typical traffic patterns in data centers.

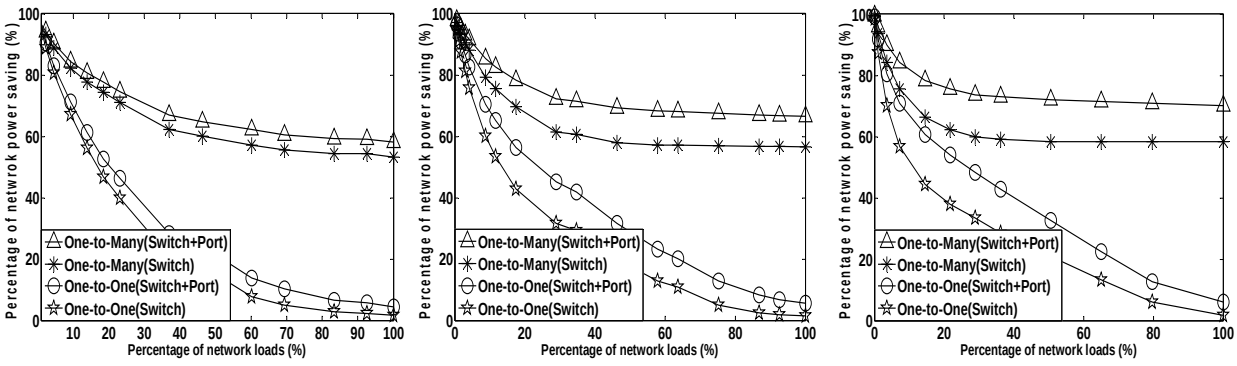
Fig. 9 and 10 show the power consumption of network devices reduced by our algorithm under One-to-One and One-to-Many traffic patterns in BCube and Fat-Tree topologies respectively. We observe that our routing algorithm can save abundant power consumed by network devices under One-to-One and One-to-Many traffic patterns under the light network loads. Furthermore, we find that the power saving effect under One-to-Many is better than that under One-to-One, when their network loads are the same. This is because our algorithm has more opportunities to aggregate network flows under the One-to-Many traffic pattern. We take BCube(8,3) and Fat-Tree(48) as examples. When the percentage of network loads is 50%, we can save about 25% of total network power under One-to-One and save 76% under the One-to-Many traffic pattern based on the "component-level" granularity in Fig. 9(c). And the power-saving percentage values can reach 33% and 72% respectively under the same load percentage in Fig. 10(c).

Fig. 11 shows the power-saving effect by our algorithm under the All-to-All traffic pattern in BCube and Fat-Tree topologies. Our algorithm only saves little power when a high performance threshold percentage is set. That is because heavy network loads reduce the opportunities of conserving power consumed by network devices. The



(a) BCube(4,2) (b) BCube(8,2) (c) BCube(8,3)

Fig. 9. Network power conservation percentage based on "device-level" and "component-level" granularities under One-to-One and One-to-Many traffic patterns in BCube topologies.



(a) FP=12 (b) FP=24 (c) FP=48

Fig. 10. Network power conservation percentage based on "device-level" and "component-level" granularities under One-to-One and One-to-Many traffic patterns in Fat-Tree topologies.

simulation results are similar in the topologies of BCube(8,3) and Fat-Tree(48), which are not drawn in Fig. 11(a) and 11(b).

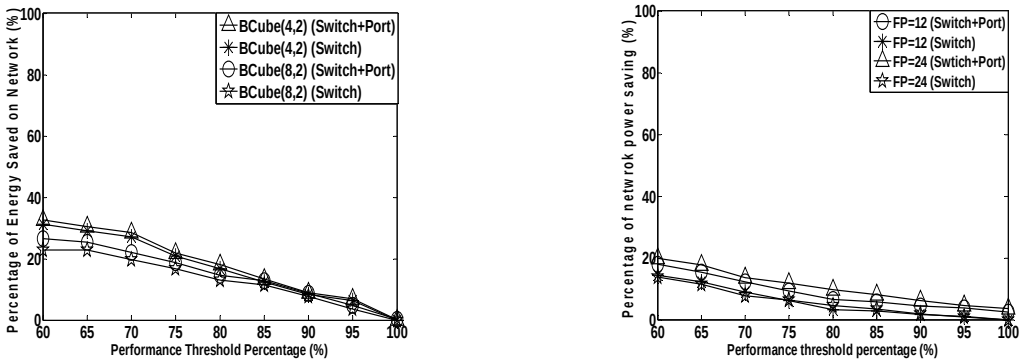
From the results above, we can summarize the power saving effectiveness of our algorithm under four traffic patterns as the following inequality:

One-to-Many > Random > One-to-One > All-to-All.

Consequently, our algorithm can save the maximum amount of power under the One-to-Many traffic pattern. With the same network load, the power saving effect under random traffic is in a range between that under One-to-One and One-to-Many traffic patterns. Moreover, our algorithm can save little power under the All-to-All traffic pattern due to heavy network loads.

4.5. Tradeoff between Power Conservation and Reliability Requirements

We evaluate the power conservation level of our algorithm under different reliability requirements using the BCube and Fat-Tree topologies. We set the reliability parameter *RP* to different values according to the characteristics of these two topologies.



(a) BCube(4,2) and BCube(8,2) (b) Fat-Tree(12 and 24)

Fig. 11. Network power conservation percentage based on "device-level" and "component-level" granularities under the All-to-All traffic pattern in BCube and Fat-Tree topologies.

Fig. 12(a) shows the network power consumption percentage saved by our algorithm under four different reliability requirements in a given BCube topology, i.e., BCube(8,3). As the BCube(8,3) topology has 4 available paths between any two servers, we vary the values of *RP* from 1 to 4. The curve marked with "*RP*=1" shows the power saving percentage with the "component-level" power-aware routing strategy but not considering any reliability requirement, which is consistent with that in Fig. 6(C) and shown here as the baseline value compared with others. The curve marked with "*RP*=4" denotes the power consumption reduced by the power-aware routing under the highest reliability restriction. As expected, the power conservation effect decreases as the values of *RP* increases under the same network load. For example, when the percentage of network loads is 20%, the power conservation percentages are about 39% and 23% when RP is 2 and 3 respectively, and approaches 10% when RP increases to 4. The difference in power conservation among the four reliability levels reduces when the network has very low load and most devices are kept idle.

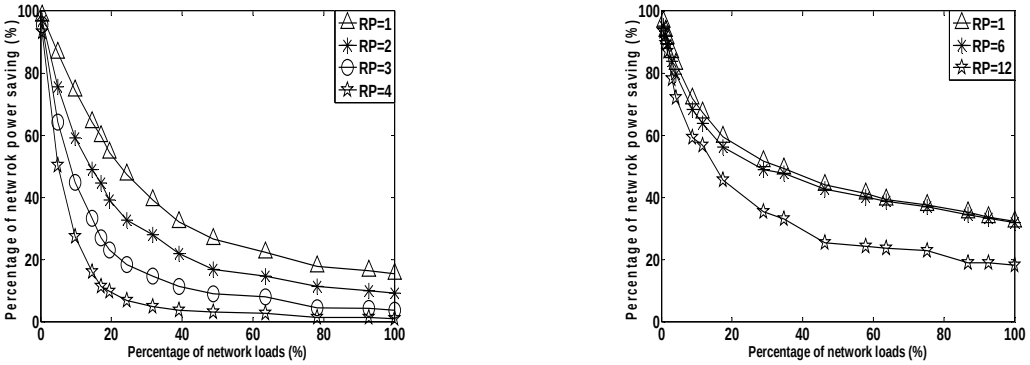
Similar results can be found in the Fat-Tree topology in Fig. 12(b). The Fat-Tree(24) topology has 12 available paths between any two intra-pod servers and 144 available paths between any two inter-pod servers. We set *RP* as 1, 6, 12 to denote the different reliability requirements in it. One important observation is that our algorithm can meet a higher reliability requirement by increasing only a few additional switches and network ports in the Fat-Tree topology. For example, we can achieve the reliability requirement "*RP*=6" from "*RP*=1", when we add about 15 switches to the topology from the 276 switches eliminated by the SE module, which only decreases power conservation percentage from 51% to 49% under 30% of network loads. It is because we can easily obtain more available paths between servers by putting back only a few additional aggregation and core switches in the Fat-Tree topology.

Similar simulation results can be also obtained in the other sizes of the BCube and Fat-Tree topologies. From the simulation results above, we can see that there is a tradeoff between the power conservation and the network reliability requirement. Our algorithm can adapt to the different reliability requirements by setting *RP* to different values.

4.6. Comparing PRA with other power conservation schemes

In this subsection, we first compare the power conservation effectiveness of our algorithm with the ElasticTree scheme in [7] as well as the optimal solution of the throughput-guaranteed power-aware routing problem. Then, we analyze the computation time of our algorithm and the optimal solution.

As presented before, the problem model in the paper is different from the ElasticTree scheme. In [7], each flow has a fixed transfer rate which is taken as the input of the power minimization problem. In contrast, the transfer



(a) BCube(8,3) (b) Fat-Tree(24)

Fig. 12. Network power conservation percentage based on the "component-level" granularity under different reliability requirements in BCube(8,3) and Fat-Tree(24) topologies.

rate of each flow in our model depends on the number of competing flows in the network. In order to make an effective and fair comparison, we first use our algorithm PRA to compute the power-aware routing as well as the corresponding transfer rate of flows, and then run the ElasticTree algorithm based on the same traffic matrix and transfer rate of flows with PRA. The purpose of this simulation setup is to compare the power conservation effectiveness of the two schemes when they can achieve the same network throughput. Moreover, as the topology-aware heuristic algorithm in [7] is used only for the Fat-Tree topology, we choose the greedy bin-packing algorithm in [7] and compare it with our PRA in BCube and Fat-Tree topologies respectively. In our following comparisons, we randomly generate the traffic matrix between servers and set the performance threshold percentage *PR* and reliability parameter *RP* as 95% and 1 respectively.

Fig. 13 and 14 show the power saving effectiveness of PRA and ElasticTree with the increase of network loads in BCube and Fat-Tree topologies respectively. From Fig. 13, we observe that our PRA can achieve better power conservation than ElasticTree in the BCube topology. For example, the power saving percentage in BCube(4,2) can be improved from 55% in the case of using ElasticTree to 63% when using PRA under the 30% network load, and the percentage figure rises from 20% with ElasticTree to 30% with PRA when the network load percentage is 80%. Similar results are also found in BCube(8,2). The curves of ElasticTree and PRA in Fig. 14 are close with each other under the same topology size, which indicates ElasticTree and PRA have similar power conservation effectiveness in the Fat-Tree topology. Also, we observe that PRA has a slightly higher power saving percentage than ElasticTree in Fat-Tree(FP=12) topology.

In Fig. 15, we compute the optimal solution of energy-aware routing by searching the space of all possible routing paths for flows and the computational complexity is $O(k^{|F|})$ and $O(k^{2|F|})$ for a *k*-level BCube topology and a *k-ary* Fat-Tree topology respectively, where $|F|$ is the number of network flows in DCNs. We select small-sized topologies: BCube(4,2) and Fat-Tree(FP=4) to compute the optimal throughput-guaranteed power-aware routing under different number of flows and compare its power conservation effectiveness with our PRA. The results in Fig. 15 indicate that there is no significant power saving gap between PRA and the optimal solution, especially when the number of flows is small. As the number of flows increases from 4 to 20, the difference of power saving percentage is at most 9% between PRA and the optimal solution in both BCube and Fat-Tree topologies. This demonstrates the efficiency of our proposed routing algorithm.

We analyze and compare the routing computation time of our PRA and the optimal solution by conducting simulations using a 2.8GHZ Intel Core2 Duo E7400 CPU with 2GB RAM. The results indicate that computing the optimal solution of power-aware routing might take about 10-20 hours in BCube(4,2) and Fat-Tree(FP=4) topologies when the number of flows is 20. Obviously, computing the optimal solutions is not practical for a real data center network due to the extremely high computational complexity. In contrast, our PRA only takes several

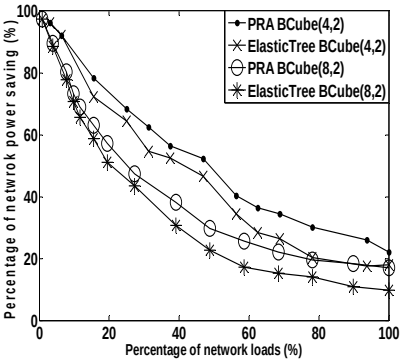


Fig.13 Comparison of PRA with ElasticTree in BCube(4,2) and Bcube(8,2) topologies under the same traffic demand

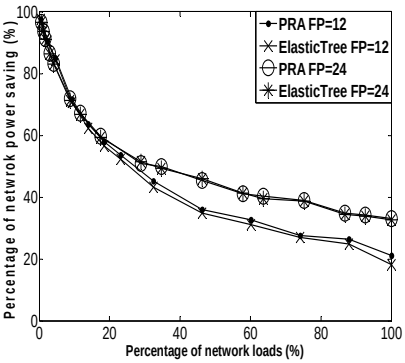


Fig.14 Comparison of PRA with ElasticTree in Fat-Tree(12 and 24) topologies under the same traffic demand

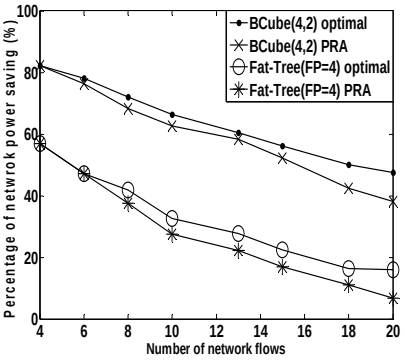


Fig.15 Comparison of PRA with the optimal solution in BCube(4,2) and Fat-Tree(4) topologies under the same traffic demand

milliseconds to obtain the power-aware routing in this case. We show the detailed routing computation time of our PRA under different network loads with BCube and Fat-Tree topologies in Fig. 16. The results indicate that our PRA only takes less than one second to find the power-aware routing paths for hundreds of flows and several seconds for thousands of flows under the medium-sized BCube and Fat-Tree topologies.

An alternative for solving the power-aware routing problem is to use the branch and bound framework or the genetic algorithm to calculate a near optimal solution. However, the method cannot meet the real-time demand of power-aware routing computation, either. Our simulation results indicate it might take several hours to obtain the near optimal solution under the light network load even in the small-sized BCube and Fat-Tree topologies.

5. Practical Issues

In this section, we discuss some practical issues when implementing our throughput-guaranteed power-aware routing scheme, including how to execute and deploy the modules of our algorithm in a real system, how to mitigate flow disruption and packet loss during the power mode transition of switches and links, and the time granularity of running the algorithm.

When implementing our algorithm in a real DCN, we first need to gather the information of network topology and traffic, which can be obtained with some existing network protocols and technologies, such as SNMP or OpenFlow [7]. The performance threshold and reliability restriction parameters can be set according to the requirements of the data center operators. Then, five modules cooperatively run on a management server and finally output the power-aware routing paths and update the forwarding tables of the network devices. The dynamical forwarding table configuration can be enabled in modern network infrastructure for data centers, such as OpenFlow framework.

It is important to avoid flow disruption and packet loss when updating the power-aware routing. In the design of SE and LE modules of our algorithm, we have restricted the rule of elimination so that any critical switch or link will not be eliminated to avoid interrupting flows in the traffic matrix. Also, we should ensure that the power mode transition of network devices and links will not bring packet loss. In the current technology, we base our design on the make-before-break technique. Active switches on the old path will not be turned off until the new path has been established and existing packets have been transmitted, while inactive switches will be brought to action before a new path is established and goes through it. We can also apply source routing schemes to avoid routing loops and black holes caused by inconsistent routing tables of network devices in DCNs. Moreover, the power conservation effect will also be affected by the power mode transition delay of network devices and links, which cannot be neglected under the current hardware technologies. Recently study shows that the power

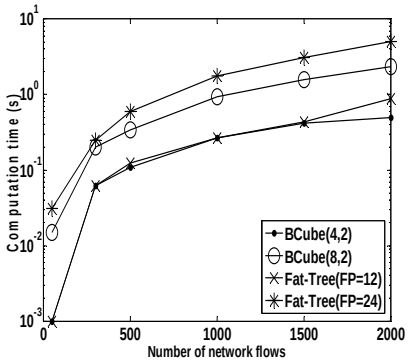


Fig. 16 Computation time of PRA for different numbers of network flows in the BCube and Fat-Tree topologies

transition speed in hardware can be as fast as milliseconds [18]. Although there are some software state processes and the actual transition time can be longer, we expect that future power management technology can realize power state transition on network devices in real time.

Our routing algorithm runs periodically so that the routing paths will be adapted when there are dramatic load changes. When the dominant communication loads between servers in a data center do not vary rapidly or can be predicted according to the historical traffic information, the updated period can be set relatively long, such as 10-30 minutes. If traffic changes are more frequent in DCNs, we should reduce the re-computation period of power-aware routing, such as 5 minutes. Moreover, we have to restrict the transition frequency between active and sleep modes of switches to avoid the network oscillation when the traffic changes dramatically.

We also consider the subsequent configuration reachability of network devices while choosing power-aware paths for flows. Our algorithm takes the network power conservation as a major objective and preferentially chooses the minimal power consumption path for each flow. If there are multiple such paths, we will further consider the power mode transition cost of switches and select the path traversing the minimal number of the sleeping switches in the last configuration, so that the number of switches transiting from the sleep mode to the active mode is as few as possible. There is a tradeoff between conserving more power and making the subsequent configuration more easily reachable.

6. Conclusion

In this paper, we address the power-saving problem in data center networks from a routing perspective. We first introduce network power consumption models, and then establish the model of throughput-guaranteed power-aware routing problem and prove that it is NP-hard by reducing 0-1 Knapsack problem into it. Based on these models, we propose a routing algorithm to solve the throughput-guaranteed power-aware routing problem. The evaluation results demonstrate that our power-aware routing algorithm can effectively conserve the power consumption of network devices in data centers, especially when the network load is low.

In the future, we would like to extend throughput-guaranteed power-aware routing to more general application contexts, e.g., delay-sensitive applications. We plan to design a distributed power-aware routing algorithm in data center networks as well. Furthermore, another interesting study direction is green virtualization technologies, such as power-aware virtual machine placement and migration strategies.

Appendix

NP Hardness Proof of PRP

To prove the NP-hardness of the throughput-guaranteed power-aware routing problem, we simplify the problem by setting both the power consumption of switch ports *A* and the predefined threshold of network reliability *RP* as zero (i.e., without considering the network reliability constraint). The objective of the simplified PRP is to compute power-aware routing paths for network flows, so that as few switches as possible are involved in the routing paths while satisfying a predefined network throughput threshold. We translate it into an equivalent problem: PRP-T.

PRP-T: For a given set of input parameters of (*G*, *T*, *K*, *N*), *G*, *T*, *K* denote topology, traffic matrix and predefined threshold of network throughput respectively, which are the same as PRP. *N* is the upper bound of number of switches. The objective of PRP-T is to find a routing *R2* for *T*, satisfying the two conditions as follows:

$$L(R2) \le N \tag{11}$$

$$M(R2) \ge K \tag{12}$$

where the functions *L*(.) and *M*(.) are the same as those in PRP.

The translation process from PRP to PRP-T is shown in the pseudocode in Fig. 17. It shows the solution of PRP can be obtained by solving PRP-T repeatedly for at most *N* steps, so the hardness of PRP is not higher than PRP-T, i.e. $\text{PRP} \le_P \text{PRP-T}$. On the other hand, obviously $\text{PRP-T} \le_P \text{PRP}$. Therefore, the hardness of PRP and PRP-T are equivalent. PRP is NP-hard if and only if PRP-T is NP-hard. Next, we present the proof of the NP-hardness of PRP-T.

Simplified Power-aware Routing Problem: PRP(*G, T, K*)

Input: *G*: DCN topology, *T*: traffic matrix, *K*: network performance threshold
begin
  set *N* := the number of all switches in the network;
  While (the solution of PRP-T(*G, T, K, N*) exists)
  begin
    set result := PRP-T (*G, T, K, N*);
    set *N* := *N*-1;
  end
  return *result*;
end

Fig. 17. Translation from PRP to PRP-T

Our idea is to reduce the classical 0-1 Knapsack problem into the PRP-T problem. 0-1 Knapsack problem belongs to the well known Karp's 21 NP-complete problems [9]. The definition of 0-1 Knapsack problem is as follows [10].

There are *n* kinds of items denoted by $W_1, W_2, ..., W_n$, and let $W = \{W_1, W_2, ..., W_n\}$. Each item W_j has a nonnegative weight S_j and a nonnegative value V_j. The maximum capacity of the bag is *C*, and *E* is defined as the predefined lower bound of the total value of items, where *C* and *E* are both nonnegative. The value of X_j restricted to 0 or 1 denotes the number of item W_j put into the bag. The object of 0-1 Knapsack problem is to find a subset of items (equivalent to assigning value to each X_j), subject to the following two conditions:

$$\sum_{j=1}^{n} S_j X_j \le C \tag{13}$$

$$\sum_{j=1}^{n} V_j X_j \ge E,\ X_j \in \{0,1\}. \tag{14}$$

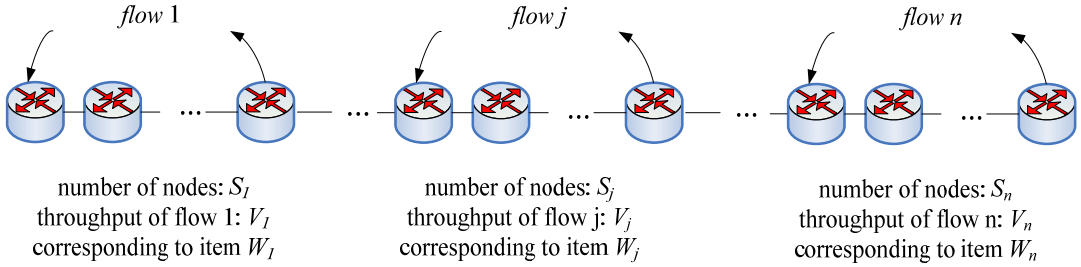


Fig. 18. Topology and traffic construction in the instance of PRP-T.

The 0-1 Knapsack problem can be reduced to PRP-T problem in polynomial time, which consists of three steps as follows.

Step 1: Instance Construction

We first construct a specific topology *G* in PRP-T. The set of all nodes in *G* is divided into *n* groups. Each group *j* contains S_j nodes and S_j-1 links, as shown in Fig. 18. The network capacity of each link in group *j* is V_j. Then we construct *n* flows: *flow* 1, *flow* 2, …, *flow* n. The source node of flow *j* is the first node in group *j*, and the destination is the last node in group *j*. Let T_j be the set of nodes in the path of flow *j*, and $T = \{T_1, T_2, ..., T_n\}$. Therefore, the number of nodes in T_j is equal to S_j and the throughput of flow *j* is equal to V_j. Finally, we let the predefined threshold *N* be equal to *C*, and let the predefined threshold *K* be equal to *E*.

Step 2: if the solution of 0-1 Knapsack problem exists, then the solution of the PRP-T instance also exists.

Proof:

If the solution of 0-1 Knapsack problem exists, i.e., $\exists W' \subseteq W$, such that

$$\sum_{W_j \in W'} S_j \le C \tag{15}$$

$$\sum_{W_j \in W'} V_j \ge E, 1 \le j \le n. \tag{16}$$

Then $\exists T' \subseteq T$, such that

$$\sum_{T_j \in T'} S_j \le N \tag{17}$$

$$\sum_{T_j \in T'} V_j \ge K, 1 \le j \le n. \tag{18}$$

Therefore, there exists the specific routing paths for n flows, so that the number of nodes involved in the routing paths is not more than *N*, and the total throughput is not less than *K*. Consequently, the solution of the PRP-T instance exists.

End.

Step 3: if the solution of the PRP-T instance exists, then the solution of 0-1 Knapsack problem also exists.

Proof:

If the solution of the PRP-T instance exists, i.e., there exists the specific routing paths for *n* flows, so that the number of nodes involved in the routing paths is not more than *N*, and the total throughput is not less than *K*. Let *NS* denote the set of nodes involved in the routing paths, let the function Tht(*NS*) denote the total throughput of the flows only traversing the nodes in *NS*, and let |*NS*| denote the number of nodes in the set *NS*. Then

$$| NS | \le N \tag{19}$$

$$\mathrm{Tht}(NS) \ge K. \tag{20}$$

We divide *NS* into two subsets: *NS*1 and *NS*2, subject to

$$NS = NS1 \bigcup NS2 \tag{21}$$

$$NS1 \bigcap NS2 = \varnothing. \tag{22}$$

*NS*1 and *NS*2 are defined by

$$NS1 = \bigcup T_j, T_j \subseteq NS, 1 \le j \le n \tag{23}$$

$$NS2 = NS - NS1. \tag{24}$$

Therefore, we can see that

$$\mathrm{Tht}(NS2) = 0, \tag{25}$$

as the nodes in *NS*2 are not able to form a complete path for any flow, and

$$\mathrm{Tht}(NS) = \mathrm{Tht}(NS1) + \mathrm{Tht}(NS2) = \mathrm{Tht}(NS1) \ge K. \tag{26}$$

On the other hand,

$$|NS1| \le |NS| \le N, \tag{27}$$

as *NS*1 is the subset of *NS*. Therefore we can find the subset of items *W'*, such that

$$\sum_{W_j \in W'} S_j = |NS1| \le C \tag{28}$$

$$\sum_{W_j \in W'} V_j = \mathrm{Tht}(NS1) \ge E. \tag{29}$$

Consequently, the solution of 0-1 Knapsack problem exists.

End.

Based on the three-step proof above, we conclude that PRP-T is an NP-hard problem. Moreover, given a routing for *T*, we can easily verify whether it can satisfy the two restrictions of PRP-T in polynomial time. Therefore, PRP-T is an NP-complete problem and then PRP is NP-hard.

References

[1] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In OSDI, 2004.
[2] S. Ghemawat, H. Gobioff, and S. Leung. The Google File System. In SOSP, 2003.
[3] CloudStore. Higher Performance Scalable Storage. http://kosmosfs.sourceforge.net/.
[4] M. Al-Fares, A. Loukissas, and A. Vahdat. A Scalable, Commodity Data Center Network Architecture. In SIGCOMM, 2008.
[5] 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. In ACM SIGCOMM, August 2009.
[6] 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, pages 75–86, 2008.
[7] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, N. McKeown. ElasticTree: Saving Energy in Data Center Networks. In NSDI'10, Apr 2010.
[8] J. G. Koomey. Growth in data center electricity use 2005 to 2010. Analytics Press, Aug. 2011.
[9] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller and J.W. Thatcher (Eds.), Complexity of Computer Computations. Plenum Press, New York, 1972.
[10] A. Levitin. Introduction to the design & analysis of algorithms. Addison-Wesley, 2003.
[11] D. Nace, N. L. Doan, E. Gourdin, B. Liau. Computing Optimal Max-Min Fair Resource Allocation for Elastic Flows. In IEEE/ACM Transactions on Networking 16(6): 1272-1281, 2006.
[12] D. Bertsekas, R. Gallager. Data networks. Englewood Cliffs, NJ: Prentice-Hall, 1992.
[13] R. N. Mysore, et al. PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric. In ACM SIGCOMM, August 2009.
[14] A. Greenberg, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta. VL2: A Scalable and Flexible Data Center Network. In ACM SIGCOMM, August 2009.
[15] D. Li, C. X. Guo, H. T. Wu, K Tan, Y. G. Zhang, S. W. Lu. FiConn: Using Backup Port for Server Interconnection in Data Centers. In INFOCOM, 2009.
[16] M. Gupta, S. Singh. Greening of the Internet. In ACM SIGCOMM, Karlsruhe, Germany. August 2003.

[17] M. Gupta and S. Singh. Using Low-Power Modes for Energy Conservation in Ethernet LANs. In INFOCOM'07, May 2007.
[18] S. Nedevschi, L. Popa, G. Iannaccone, etc. Reducing Network Energy Consumption via Sleeping and Rate-Adaptation. In Proceedings of the 5th USENIX NSDI, pages 323–336, 2008.
[19] K. Christensen, B. Nordman, R. Brown. Power Management in Networked Devices. In IEEE COMPUTER SOCIETY, August 2004.
[20] J. Chabarek, J. Sommers, P. Barford, et al. Power Awareness in Network Design and Routing. In INFOCOM'08, Apr 2008.
[21] G. Magklis, M. Scott, G. Semeraro, and etc. Profile-based Dynamic Voltage and Frequency Scaling for a Multiple Clock Domain Microprocessor. In ISCA'03, Jun 2003.
[22] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti. Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures. Communications Surveys Tutorials, IEEE, 2011.
[23] G. Ananthanarayanan and R. H. Katz. Greening the Switch. In HotPower'08, Dec 2008.
[24] C. Patel, C. Bash, R. Sharma, M. Beitelmam, and R. Friedrich. Smart Cooling of data Centers. In Proceedings of InterPack, July 2003.
[25] S. Nedevschi, J. Chandrashekar, J. Liu, B. Nordman, S. Ratnasamy and N. Taft. Skilled in the Art of Being Idle: Reducing Energy Waste in Networked Systems. In NSDI'09, Apr 2009.
[26] S. Srikantaiah, A. Kansal and F. Zhao. Energy Aware Consolidation for Cloud Computing. In HotPower'08, Dec 2008.
[27] M. Pervila, J. Kangasharju. Running Servers around Zero Degrees. In ACM SIGCOMM Workshop on Green Networking 2010.
[28] P. Mahadevan, P. Sharma, S. Banerjee, P. Ranganathan. A Power Benchmarking Framework for Network Devices. In Proceedings of IFIP Networking, May 2009.
[29] P. Mahadevan, S. Banerjee, P. Sharma. Energy Proportionality of an Enterprise Network. In ACM SIGCOMM Workshop on Green Networking 2010.
[30] Y. F. Shang, D. Li, M. W. Xu. Energy-aware routing in data center network. In ACM SIGCOMM Workshop on Green Networking 2010.
[31] W.Fisher, M.Suchara, J. Rexford. Greening Backbone Networks: Reducing Energy Consumption by Shutting Off Cables in Bundled Links. In ACM SIGCOMM Workshop on Green Networking 2010.
[32] Cisco Catalyst 2960 series switches data sheet. http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps6406/product_data_sheet0900aecd80322c0c.html
[33] L. Liu, H. Wang, X. Liu, X. Jin, W. He, Q. Wang, Y. Chen. GreenCloud: A New Architecture for Green Data Center. In 6th international conference industry session on Autonomic computing and communications, 2009.
[34] Cisco Data Center Infrastructure 2.5 Design Guide. http://www.cisco.com/application/pdf/en/us/guest/netsol/ns107/c649/ccmigration_09186a008073377d.pdf, December 2007
[35] L. A. Barroso and U. Hlzle. The case for energy-proportional computing. In Computer, 40(12):33–37, 2007.
[36] A. P. Bianzino, C. Chaudet, D. Rossi, and J.-L. Rougier. A Survey of Green Networking Research. In IEEE Communications Surveys & Tutorials, 14(1): 3-20, 2012.
[37] M. Zhang, C. Yi, B. Liu, B. Zhang. GreenTE: Power-Aware Traffic Engineering. In Proceedings of the 18th IEEE International Conference on Network Protocols (ICNP' 10), page 21-30, 2010.
[38] N. Vasic, D. Kostic. Energy-Aware Traffic Engineering. In Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking (e-Energy'10), page 169-178, 2010.
[39] A. Cianfrani, V. Eramo, M. Listanti, M. Marazza, E. Vittorini. An Energy Saving Routing Algorithm for a Green OSPF Protocol. In Proceedings of the IEEE INFOCOM 2010 Workshop on Computer Communications, page 1-5, 2010.
[40] A. Cianfrani, V. Eramo, M. Listanti, M. Polverini. An OSPF Enhancement for energy saving in IP Networks. In Proceedings of the IEEE INFOCOM 2011 Workshop on Computer Communications, page 325-330, 2011.
[41] R. McGeer, P. Mahadevan, S. Banerjee. On the Complexity of Power Minimization Schemes in Data Center Networks. In GLOBECOM'10, Dec, 2010.
[42] P. X. Gao, A. R. Curtis, B. Wong, S. Keshav. It's Not Easy Being Green. In SIGCOMM, 2012.
[43] N. Vasic, P. Bhurat, D. Novakovic, M. Canini, S. Shekhar, and D. Kostic, Identifying and using energy-critical paths. In Proceedings of the CoNEXT, 2011.
[44] Y. M. Kim, E. J. Lee, H. S. Park, J. K. Choi, H. S. Park. Ant colony based self-adaptive energy saving routing for energy efficient Internet. In Computer Networks, 56(10): 2343-2354, 2012.
[45] F. Cuomo, A. Cianfrani, M. Polverini, D. Mangione. Network pruning for energy saving in the Internet. In Computer Networks, 56(10): 2355-2367, 2012.
[46] S. Avallone, G. Ventre. Energy efficient online routing of flows with additive constraints. In Computer Networks, 56(10): 2368-2382, 2012.
[47] W. Hou, L. Guo, X. Wei, X. Gong. Multi-granularity and robust grooming in power- and port-cost-efficient IP over WDM networks. In Computer Networks, 56(10): 2383-2399, 2012.
[48] J. L. Vizcaino, Y. Ye, I. T. Monroy. Energy efficiency analysis for flexible-grid OFDM-based optical networks. In Computer Networks, 56(10): 2400-2419, 2012.
[49] S. Ricciardi, F. Palmieri, U. Fiore, D. Careglio, G. Santos-Boada, J. Sole-Pareta. An energy-aware dynamic RWA framework for next-generation wavelength-routed networks, In Computer Networks, 56(10): 2420-2442, 2012.
[50] G. Rizzelli, A. Morea, M. Tornatore, O. Rival. Energy efficient Traffic-Aware design of on-off Multi-Layer translucent optical networks. In Computer Networks, 56(10): 2443-2455, 2012.
[51] K. K. Nguyen, et al. Environmental-aware virtual data center network. In Computer Networks, 56(10): 2538-2550, 2012.
[52] L. Gyarmati, T. A. Trinh. How can architecture help to reduce energy consumption in data center networking? In Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking (e-Energy'10), pages 183-186, 2010.

Mingwei Xu received the B.S. degree in 1994 and Ph.D. degree in 1998 both from the Department of Computer Science and Technology, Tsinghua University, Beijing, China. He is currently a professor in Tsinghua University. His research interests include computer network architecture, Internet protocol and routing, high-speed router architecture and green networking.

Yunfei Shang received the B.S. and M.S. degrees from the Department of Computer Science and Technology, National University of Defense Technology in 2003 and 2006 respectively. Now, he is a Ph.D. candidate in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research interests include Internet architecture, green networking, data center network, network routing.

Dan Li received the M.E. degree and Ph.D from Tsinghua University in 2005 and 2007 respectively, both in computer science. Before that, he spent four undergraduate years in Beijing Normal University and got a B.S. degree in 2003, also in computer science. He joined Microsoft Research Asia in Jan 2008, where he worked as an associate researcher in Wireless and Networking Group until Feb 2010. He joined the faculty of Tsinghua University in Mar 2010, where he is now an Assistant Professor in Computer Science Department. His research interests include Internet architecture and protocol design, P2P networks, cloud computing networks and green networking.

Xin Wang received the B.S. and M.S. degrees in telecommunications engineering and wireless communications engineering from Beijing University of Posts and Telecommunications, Beijing, China, and the PhD degree in electrical and computer engineering from Columbia University, New York, NY. She is currently an assistant professor in the department of Electrical and Computer Engineering of the State University of New York at Stony Brook, Stony Brook, New York. Before joining Stony Brook University, she was a Member of Technical Staff in the area of mobile and wireless networking at Bell Labs Research, Lucent Technologies, New Jersey and an Assistant Professor in the Department of Computer Science and Engineering of the State University of New York at Buffalo, Buffalo, New York. Her research interests include analysis and architecture design in wireless networks and communications, mobile and distributed computing, infrastructure design and performance enhancement across network layers, applications and heterogeneous networks, network and mobility management, QoS, signaling and control, as well as support for advanced network services and applications. Dr. Wang has been a member of the Association for Computing Machinery (ACM) since 2004.