Future Generation Computer Systems 91 (2019) 25–36

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Comprehensive link sharing avoidance and switch aggregation for software-defined data center networks

Yue Zeng, Songtao Guo *, Guiyan Liu

National & Local Joint Engineering Laboratory of Intelligent Transmission and Control Technology (Chongqing), College of Electronic and Information Engineering, Southwest University, Chongqing, 400715, China

HIGHLIGHTS

- We formulate the network energy consumption minimization problem under the constraints of traffic conservation, link capacity and state relation.
- We propose the link sharing avoidance algorithm from time dimension as well as the switch aggregation algorithm from power dimension to achieve energy saving.
- We propose a heuristic integrated time and power (ITP) algorithm to further reduce energy consumption.
- Experiment results show that our proposed algorithm has more energy saving and lower network delay for different network topologies, network scales, flow sizes and flow arrival rates.

ARTICLE INFO

Article history:
Received 6 April 2018
Received in revised form 15 July 2018
Accepted 19 August 2018
Available online xxxx

Keywords:
Link sharing avoidance
Switch aggregation
Energy saving
Software defined data center networks
Heuristic algorithm

ABSTRACT

An effective way to reduce network energy consumption of data center networks (DCNs) is to activate network elements as few as possible, complete transmission in as short a time as possible, and set unnecessary network elements to sleep mode. At present, most existing energy saving works considered the network energy saving from the dimension of time or power separately. However, in fact these two dimensions can interact with each other, i.e., reducing the network delay may lead to the increase of network energy consumption, and vice versa. In this paper, two dimensions of time and power are comprehensively studied in the Minimum Network Energy Consumption (MNEC) problem. First of all, we formulate the MNEC problem by considering both time and power, and prove that it is a NP-hard problem. Furthermore, we propose a heuristic Integrated Time and Power (ITP) algorithm, which combines the link sharing avoidance algorithm to reduce the network delay from the dimension of time as well as the switch aggregation algorithm to reduce the energy consumption from the power dimension. Finally, the performance of ITP algorithm is evaluated under different network topology, network size, traffic size and flow number under the network environment based on Mininet and Ryu controller. Experimental results show that the ITP algorithm outperforms the existing network energy saving algorithm in terms of energy consumption.

1. Introduction

In recent years, with the development of cloud computing and video services, the scale and energy consumption of data centers have increased dramatically. In 2013, the energy consumption of data centers in the United States reached 91 billion kilowatt-hours, which was close to that of the Three Gorges hydropower station, the largest hydropower station in the world, with an annual capacity of 98 billion kilowatt-hours. In addition, it is expected that by 2020 the energy consumption will reach 140 billion kilowatt-hours [1]. Due to the rapid increase of energy consumption in data centers, energy saving has become one of the major bottlenecks of design data centers. In the meanwhile, the network energy consumption dominates the whole energy consumption in the data center, up to 15% [2]. Therefore, how to reduce data center network energy has become a hot topic in industry and academia.

In order to ensure that the data center has better performance, researchers have proposed numerous the network structures with redundant links, such as Fat-Tree [3], Bcube [4], FiConn [5]. Although redundant links and switches in network topology designed for meeting peak traffic requests can improve the performance of network bandwidth, they also lead to a sharp increase in network energy consumption since peak traffic is not regular at most of the time in the network. This means that running unnecessary links and switches will lead to the waste of network energy.

* Corresponding author.
E-mail address: stguo@swu.edu.cn (S. Guo).

https://doi.org/10.1016/j.future.2018.08.034
0167-739X/

An effective way to save energy is to activate as few switches and links as possible, finish transfers in as short time as possible, and sleep unnecessary network elements. Most existing works focus on local energy saving by considering how to make the switch component sleep for a long time when the switch is idle [6–8], or lets the switch adjust the sending rate according to network load [9]. In fact, switches can only obtain information from the surrounding switches and hosts, and cannot optimize the network energy globally in traditional networks.

Software Defined Networking (SDN) technology with separated control plane and data plane is emerged as a new network architecture. The SDN controller can obtain the global network information and control the data transmission in the whole network. Owing to the global network view, operators can collect topology information and forwarding state of the entire network. Meanwhile, the network flow can be adjusted dynamically, and more reasonable and energy-saving path for the flow is planned. Numerous network energy conservation works [10–15] studied the network energy conservation from the global view of the network. These research works can be divided into two categories. From the power dimension, part of the research works [10–14] aggregated traffic into the smallest set of network elements, and let the idle network elements be sleep mode to reduce energy consumption. However, their traffic aggregation allows link sharing, which creates more network bottlenecks and reduces link utilization. From the dimension of time, part of the research works [15,16] employed flow scheduling method to reduce the generation of network bottlenecks and network latency, so as to achieve the purpose of energy saving. However, these research works only plan a shortest path for the flow from idle links, rather than finding the path with minimum power based on the network state.

Different from the previous works of saving energy from time or power dimension separately, in this paper, we integrate two dimensions of power and time to find a path set for all traffic requests, activate links and switches in the path set, and configure unnecessary network components to sleep mode. From the time dimension, we propose a link sharing avoidance algorithm to improve the link utilization, avoid the network bottleneck and reduce the network transmission delay by making the flow occupy the link separately, so as to reduce the network energy consumption. In this algorithm, it is worth noting that if the given flow cannot find a path to avoid the shared link, then the path with minimum energy consumption for the shared link is planned for the flow. From the dimension of power, we propose a switch aggregation algorithm that improves the utilization of switches and reduces energy consumption without link sharing. By combining the link sharing avoidance algorithm with the switch aggregation algorithm, furthermore, we propose a heuristic Integrated Time and Power (ITP) algorithm to transform the path search problem of minimum network energy consumption into the minimum weight path planning problem. The link sharing avoidance algorithm and the switch aggregation algorithm jointly determine the path selection by setting the appropriate path weight. The weight of a path reflects the energy consumed in the path, that is, the smaller the weight, the less energy consumption. As a result, the time complexity of the minimum energy path planning problem is reduced to the time complexity of the minimum weight path search problem, so that the algorithm has a high real-time performance and can be deployed in large-scale networks.

The main contributions of this paper are summarized as follows:

- We formulate the Minimum Network Energy Consumption (MNEC) problem under the constraints of flow conservation, link capacity and state relation. Then, we prove that it is a NP-hard problem and transform it into the minimum weighted path planning problem in the weighted graph.
- We propose the link sharing avoidance algorithm from time dimension as well as the switch aggregation algorithm from power dimension to achieve energy savings. In particular, we propose a heuristic Integrated Time and Power (ITP) algorithm to further reduce energy consumption.
- We evaluate our algorithm in the experimental environment built by Mininet [17] and Ryu controller [18]. Experimental results show that our proposed algorithm has more energy savings under different network topologies, network scales, flow sizes and the number of flows.

The reset of the paper is organized as follows. In Section 2, we introduce the related work of network energy saving. In Section 3, we establish the MNCP model and proved that it is a NP-hard problem. Then, we propose the ITP algorithm that combines two dimensions of time and power to save network energy consumption in Section 4. In Section 5, the performance of our algorithm is tested through extensive simulation. Section 6 concludes this paper.

2. Related work

In this section, we will introduce the related work on energy savings in traditional data center networks (DCNs) and the emerging Software Defined Networking (SDN).

In the traditional DCNs, the switches can only obtain the information of their surrounding nodes, thus it is difficult to optimize the entire network. Therefore, numerous research efforts [6–9,19,20] focus on the optimization of local energy. Gupta et al. [6] showed that setting the switch to sleep mode saves network energy and does not affect the upper layer protocol when the device is idle. In [7], they verified that the sleep of switches in local area network (LAN) can be indeed beneficial to save network energy. As an extension, they proposed a switch DELS sleep mechanism [19], which enables the switch to take into account both packet transmission and energy saving under low load conditions. In [8], they proved that energy can be greatly reduced by turning off the switch ports, even when the network load is heavy, and the delay caused by energy saving does not affect the behavior of the high-level protocol. In addition, it is found that the switching fabric can also configured in sleep. Furthermore, Nedevschi et al. [9] demonstrated that the energy saving mechanisms based on adaptive network load and idle time sleep can reduce the energy consumption of the network.

With the emergence of SDN technology with a global view of the network, many research works [6,10,11,15,21–25] make efforts on the global optimization of the network. From the dimension of power, Gupta et al. [6] proposed to transfer the low load link traffic to other links, so that more links are idle and the idle links are set to sleep. However, this paper does not consider network layer routing algorithm. Furthermore, Chabarek et al. [21] made a survey on energy saving of network and routing design in wired networks, but they do not carry out specific routing design. Seetharaman et al. [22] proposed the Elastictree algorithm which makes full use of the Fat-Tree data center networks, but they only focus on energy saving in the tree-based network. Zhang et al. [23] proposed the GreenTE model, which is a maximum energy saving model under the constraints of link utilization and packet delay. However, the model being solved by Cplex has higher time complexity when the scale of network is larger.

To overcome excessive time consumption for energy saving in large-scale networks, Wang et al. [10] proposed two greedy algorithms to ensure a certain real-time performance. In addition, they proposed a heuristic algorithm for partially deployed SDN from the power dimension, which generates more idle links by aggregating traffics from low-load links [11]. However, it allows link sharing, which reduces power consumption, but greatly increases network

Table 1
Notations.

Symbol	Meaning
S	Set of switches
L	Set of links
E	Energy consumption of the whole network
$E_{switch}(s)$	Energy consumption of switch s
$E_{fixed}(s)$	Fixed energy consumption of switch s
$E_{ports}(s)$	Energy consumption for all ports on switch s
$E_{port}(s, i)$	Energy consumption of port i on switch s.
$E_{link}(u, v)$	Energy consumption of link (u, v)
$T_{trans}(m)$	Transmission delay of flow m
$T(s)$	Activation duration of switch s
$T(u, v)$	Activation duration of link (u, v)
$X_m(u, v)$	Binary variable indicates whether flow m is the maximum duration flow on (u, v)
$Y_m(u)$	Binary variable indicates whether flow m is the maximum duration flow on u
$Z_m(u, v)$	Binary variable indicates whether flow m passes through (u, v)
r_m	Size of flow m
b_m	Bandwidth of flow m
$f_m(u, v)$	The flow m along link (u, v)
$c(u, v)$	The capacity of link (u, v)
$a(u, v)$	Power of link (u, v)
$b(s)$	Fixed power of switch s
$((u, i), (v, j))$	Link from port i of switch u to port j of switch v
(u, v)	Link from switch u to switch v
κ_m	Flow m with source s, destination d and data size r_m
$\alpha, \beta, \gamma, \delta$	Proportional coefficient constant

latency and packet loss, thereby further increasing network energy consumption. From the dimension of time, Li et al. [15] proposed the exclusive routing to save energy, which allows the flow to occupy the link alone to reduce network bottlenecks and improve link utilization. On this basis, Xu et al. [16] improved the quality of service by taking into account the flow deadline, and also improves the link and switch utilization by preferentially scheduling the flow with the minimum activation time. In the process of path planning, they only planed the shortest path for the flow. However, the shortest path is not always the path with minimum energy. In [25], Zhao et al. verified that link sharing will greatly increase the network transmission delay and the network energy consumption in multipath routing. Different from previous work, we jointly consider two dimensions of time and power to reduce network energy consumption.

3. System model

In this section, we first introduce the energy consumption characteristics of switches and network. Subsequently, we establish the Minimum Network Energy Consumption (MNEC) problem according to the energy consumption characteristics. Finally, we prove that the MNEC problem is a NP-hard problem. The notations used are listed in Table 1.

3.1. Energy features of switch

The energy consumption of switches consists of three parts: chassis, line-cards and ports. Among them, the chassis and line-cards energy consumption are fixed. Once the switch is activated, fixed energy consumption will be generated. Ports can be configured to activate or sleep independently, resulting in dynamic changes in the energy consumption of the switch. Therefore, the switch energy consumption can be divided into two parts: ports energy consumption and fixed energy consumption. Although the energy consumption of chassis and line-cards is the main part (up to 150 watts) of the energy consumption of switch, the energy consumption of ports cannot be ignored (1–2 watts per port). In addition, increasing traffic from zero to full load will increase port power by less than 5%, which means that the impact of port capacity utilization is negligible [26]. Based on the above analysis, we can build the following energy consumption model of switch s:

$$E_{switch}(s) = E_{ports}(s) + E_{fixed}(s) \tag{1}$$

where $E_{ports}(s)$, $E_{fixed}(s)$ and $E_{switch}(s)$ indicate the ports energy consumption, fixed energy consumption and total energy consumption of the switch s, respectively. Once the switch is activated, i.e., at least one port is activated, the fixed and dynamic energy consumption are generated. Otherwise, if all ports of switch fall asleep, the switch can be set to sleep, and the switch's energy consumption is 0.

3.2. Energy features of network

The total network energy consumption consists of both fixed and dynamic energy consumption of all switches. Therefore, the network energy consumption can be expressed by

$$E = \sum_{s\in S} E_{switch}(s) = \sum_{s\in S} E_{ports}(s) + \sum_{s\in S} E_{fixed}(s) \tag{2}$$

In order to simplify the expression of network energy consumption, We replace port energy consumption with link energy consumption. Link $((u, i), (v, j))$ is simplified to (u, v). $E_{links}(u, v)$ represents the energy consumption at link (u, v), which is equivalent to the ports energy consumption at both ends of link (u, v). Thus, the network energy consumption is expressed by

$$\begin{aligned} E &= \sum_{((u,i),(v,j))\in L} (E_{port}(u, i) + E_{port}(v, j)) + \sum_{s\in S} E_{fixed}(s) \\ &= \sum_{((u,i),(v,j))\in L} E_{links}((u, i), (v, j)) + \sum_{s\in S} E_{fixed}(s) \\ &= \sum_{(u,v)\in L} E_{links}(u, v) + \sum_{s\in S} E_{fixed}(s) \end{aligned} \tag{3}$$

It is known that the energy consumption is equal to the power multiplies the time, i.e., $E = P * T$. $a(u, v)$ and $b(s)$ represent the power of link (u, v) and switch s. $T(u, v)$ and $T(s)$ represent the activation duration of link (u, v) and switch s. Thus Eq. (3) can be transformed as

$$E = \sum_{(u,v)\in L} a(u, v) * T(u, v) + \sum_{s\in S} b(s) * T(s) \tag{4}$$

where $T_{trans}(m)$ represents the transmission duration of flow m, r_m denotes the data size of flow m, and b_m indicates the bandwidth of flow m. According to the relationship among transmission duration, bandwidth and data size, we have the following equation

$$T_{trans}(m) = \frac{r_m}{b_m} \tag{5}$$

The goal of this work is to optimize the energy consumption for the transmission of data packets. Thus, we get the following equation

$$T(u, v) = \sum_{m=1}^{M} (\frac{r_m}{b_m} * X_m(u, v)) \tag{6}$$

where the term $\sum_{m=1}^{M} (\frac{r_m}{b_m} * X_m(u, v))$ represents the transmission duration on link (u, v). If flow m is the flow with maximum duration on link (u, v), then the transmission duration of flow m is equal to the transmission duration of link (u, v).

Similarly, the activation duration of switch u can be given by

$$T(u) = \sum_{m=1}^{M}(\frac{r_m}{b_m} * Y_m(u)) \tag{7}$$

where the term $\sum_{m=1}^{M}(\frac{r_m}{b_m} * Y_m(u))$ represents the transmission duration on switch u. If flow m is the flow with maximum duration on switch u, then the transmission duration of flow m is equal to the transmission duration of switch.

3.3. Energy saving problem formulation

According to the above analysis of the characteristics of the energy consumption of switches and network, we substitute Eqs. (6) and (7) into the energy expression (4), and can formulate the Minimum Network Energy Consumption (MNEC) problem as follows

$$\begin{aligned} min \sum_{(u,v)\in L}(a(u,v) * \sum_{m=1}^{M}(\frac{r_m}{b_m} * X_m(u,v))) \\ + \sum_{u\in S}(b(u) * \sum_{m=1}^{M}(\frac{r_m}{b_m} * Y_m(u))) \end{aligned} \tag{8}$$

where $\sum_{(u,v)\in L}(a(u,v) * \sum_{m=1}^{M}(\frac{r_m}{b_m} * X_m(u,v)))$ and $\sum_{u\in S}(b(u) * \sum_{m=1}^{M}(\frac{r_m}{b_m} * Y_m(u)))$ represent total energy consumption of links and total energy consumption of the switches in the network respectively.

The optimization problem (8) is constrained by the following conditions.

(1) **Flow conservation constraint:** The flow out of a switch is equal to the flow entering the switch, except for the source, which "produces" flow, and the destination, which "consumes" flow. $\sum_{\{v|(u,v)\in L\}} f_m(u,v)$ denotes the traffic generated on switch u and $\sum_{\{v|(v,u)\in L\}} f_m(v,u)$ denotes the traffic received on switch u. Moreover, we assume that there are b_m units of traffic generated at source src and b_m units of traffic received at destination dst. The traffics at immediate nodes are conserved, i.e., $\sum_{\{v|(u,v)\in L\}} f_m(u,v) - \sum_{\{v|(v,u)\in L\}} f_m(v,u)$ should be zero. Overall, the traffic conservation constraint can be given by

$$\begin{aligned} &\sum_{\{v|(u,v)\in L\}} f_m(u,v) - \sum_{\{v|(v,u)\in L\}} f_m(v,u) \\ &= \begin{cases} b_m, \text{ if } u = src \\ -b_m, \text{ if } u = dst \\ 0, \text{ otherwise} \end{cases} \end{aligned} \tag{9}$$

(2) **Link capacity constraint:**

We assume that $\sum_{m=1}^{M} f_m(u,v)$ denotes the total traffic passing through link (u,v), and $c(u,v) * Z_m(u,v)$ represents the capacity of link (u,v), where $Z_m(u,v)$ is binary variable indicating whether flow m passes through (u,v). If a flow passes through the link (u,v), the link is activated and the capacity is greater than zero. Otherwise, the link is in sleep and its capacity is zero. When the link is activated, the total traffic on the link should not exceed its capacity. Link capacity constraint can be given by

$$\sum_{m=1}^{M} f_m(u,v) \le c(u,v) * Z_m(u,v) \tag{10}$$

(3) **Bottleneck link capacity constraint:**

The bandwidth of flow depends on how many flows pass through the bottleneck link when the following transmission methods: RAPID [27], UDT [28], RUNAT [29], or TCP variants, are utilized. On the bottleneck links, the traffic is equal to the capacity. The bottleneck constraint can be expressed by

$$f_m(u,v) = \frac{c(u,v)}{\sum_{m=1}^{M} Z_m(u,v)} \tag{11}$$

where $\sum_{m=1}^{M} Z_m(u,v)$ represents the number of flows passing through link (u,v).

(4) **State relation constraint:**

The flow m with maximum duration on link (u,v) means that the flow m passes through link (u,v), and has the longest duration in all flows over link (u,v). State relation constraint on link (u,v) can be represented as

$$X_m(u,v) = \begin{cases} 1, \textit{ if } m = \text{argmax}_{\forall j}\{\frac{r_j}{b_j} * Z_j(u,v)\} \\ 0, \textit{ otherwise} \end{cases} \tag{12}$$

where the term $\frac{r_j}{b_j} * Z_j(u,v)$ represents the duration of flow j on link (u,v). If the flow j passes (u,v), then the term equals $\frac{r_j}{b_j}$, otherwise, 0.

Similarly, the flow m with maximum duration on switch u means that flow m passes through the links connected to switch u, and its duration is longest in all flows over links to switch u. State relation constraint on switch u can be given by

$$Y_m(u) = \begin{cases} 1, \textit{ if } m = \text{argmax}_{\forall j, \forall v|(u,v)\in L}\{\frac{r_j}{b_j} * Z_j(u,v)\} \\ 0, \textit{ otherwise} \end{cases} \tag{13}$$

If the traffic of flow m over (u,v) is greater than 0, it means that the flow m passes through link (u,v), which can be represented by

$$Z_m(u,v) = \begin{cases} 1, \textit{ if } f_m(u,v) > 0 \\ 0, \textit{ otherwise} \end{cases} \tag{14}$$

3.4. NP-hard proof of MNEC problem

In this subsection, we prove that the MNEC problem is a NP-hard problem. To this end, we first describe the classical 0-1 knapsack problem, and then reduce MNEC problem to the 0-1 knapsack problem.

3.4.1. 0-1 knapsack problem

The 0-1 knapsack problem can be described as: Given a set of items N and each item with a weight and a value, we determine which items are included in a collection so that the total value is as large as possible under the constraint of a given weight limit. Let v_i and w_i be the value and weight of the item i respectively. Let W be the limit of total weight and the binary variable x_i indicate whether item i is included. Thus, the 0-1 knapsack problem can be expressed as

$$\begin{aligned} &max \sum_{i=1}^{N} v_i x_i \\ &s.t. \begin{cases} \sum_{i=1}^{N} w_i x_i \le W, \\ x_i \in \{0,1\}, \quad i = 1, \ldots, N. \end{cases} \end{aligned} \tag{15}$$

Theorem 1 (*NP-hard*). *MNEC problem is a NP-hard problem.*

Proof. In order to prove that MNEC problem is a NP-hard problem, we consider a simple case with a topology as shown in Fig. 1. Suppose there is a traffic demand $\kappa = \{(s,d,r)\}$ with source s, destination d and data size r. The bandwidth of (v,d) is less than or equal to any link bandwidth from s to v. It is not difficult to

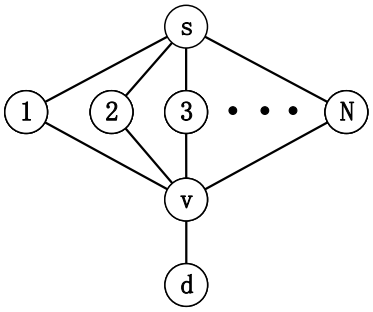


Fig. 1. Topology as an example of NP-hard proof.

observe from Fig. 1 that link (v, d) is a bottleneck link. According to bottleneck capacity constraint and flow conservation constraint, we have $b_1 = c(v, d)$ and $f_1(s, i) = f_1(i, v)$. By the state relation constraint, we have $X_1(s, i) = X_1(i, v) = Y_1(i) = Z_1(s, i) = Z_1(i, v)$. We define $Z_i \triangleq X_1(s, i) = X_1(i, v) = Y_1(i) = Z_1(s, i) = Z_1(i, v)$ and let

$$a_i = (a(s, i) + a(i, v) + b(i)) * \frac{r_1}{c(v, d)}$$

$$c_i = min\{c(s, i), c(i, d)\}$$

$$constant = (a(v, d) + b(s) + b(v) + b(d)) * \frac{r_1}{c(v, d)}$$

Furthermore, the MNEC problem can be reformulated as

$$min \sum_{i=1}^{N} a_i Z_i + constant$$

$$s.t. \begin{cases} \sum_{i=1}^{N} c_i Z_i \geq c(v, d) \\ Z_i \in \{0, 1\}, \quad i = 1, \ldots, N. \end{cases} \tag{16}$$

It is clear that problem (16) is equivalent to problem (15), which means that the MNEC problem can be reduced to the 0-1 knapsack problem. Since the 0-1 knapsack problem is a known NP-hard problem, our MNEC problem is also NP-hard. ■

4. Algorithm description

Dijkstra algorithm is the most common algorithm to find the shortest path and the minimum weight path, which has low time complexity. Considering the time dimension, the link sharing will lead to longer network delay and increase network energy consumption while port energy consumption on switches is far less than the fixed energy consumption considering the power dimension. Thus, we use the link weight to reflect the link power, thus the minimum power path finding problem is transformed into the minimum weight path finding problem. Specifically, we propose a link sharing avoidance algorithm described in Algorithm 1 to reduce network delay and energy consumption and a switch aggregation algorithm described in Algorithm 2 to reduce power by increasing the port utilization of activated switches. Both algorithms will be given in Sections 4.1 and 4.2, respectively.

In the proposed algorithms, therefore, the weight plays an imperative role in the path planning. In the following, we would like to introduce some variables that are used to determine weights in the algorithms. P_{weight} represents the energy cost of activating a port in sleep mode, which is proportional to the power of port and is expressed in Eq. (17). S_{weight} represents the energy cost of activating a sleep switch, which is proportional to the fixed power of switch and is given as Eq. (18). X_{weight} represents the cost of occupying a link that already has other flows, which is proportional to P_{weight} and S_{weight} and is given by Eq. (19). In general, to avoid sharing links, the ratio coefficients α and β are set to be a large constant.

Table 2
Comparison of energy consumption between link sharing avoidance and link sharing without avoidance.

	Link sharing avoidance	Link sharing without avoidance
Time consumption	$\frac{m}{c}$	$\frac{2*m}{c}$
Power	$8a + 24b$	$7a + 20b$
Energy consumption	$\frac{m}{c} * (8a + 24b)$	$\frac{2*m}{c} * (7a + 20b)$

$$P_{weight} = \gamma * a(u, v) \tag{17}$$

$$S_{weight} = \delta * b(u) \tag{18}$$

$$X_{weight} = \alpha * S_{weight} + \beta * P_{weight} \tag{19}$$

4.1. Link sharing avoidance algorithm

The main idea of link sharing avoidance algorithm is to avoid the link sharing by increasing the weight of the links that have been occupied by other flows. The event of triggering this algorithm has two cases, i.e., the arrival of new flows, or the completion of previous flows. If in the first case that a new flow takes a link, then we increase the weight of the link by X_{weight}. In this way, other flows will avoid using this link. If in the second case that the previous flows have been completed, then these occupied links need to be released and the link weight on the path are reduced by X_{weight}. The main procedures of the algorithm are given in Algorithm 1. It is important to note that if there is no path to avoid sharing links, a minimum energy consumption path with shared links is planned for the flow.

In order to explain the advantages of link sharing avoidance algorithm, we give examples as shown in Fig. 2. Assuming the bandwidth of each link is c, the fixed power of each switch is a and the power of each port is b. Suppose there are two flow requests: $f1(h1 \rightarrow h6), f2(h4 \rightarrow h8)$. In order to simplify the representation, we assume that the two flows have the same size, i.e., m. As shown in Fig. 2(a): without avoiding link sharing, the path planned for the two flows is: $f1(h1 \rightarrow s9 \rightarrow s5 \rightarrow s1 \rightarrow s7 \rightarrow s11 \rightarrow h6), f2(h4 \rightarrow s10 \rightarrow s5 \rightarrow s1 \rightarrow s7 \rightarrow s12 \rightarrow h8)$. As shown in Fig. 2(b): avoiding link sharing, the path of the two flows is: $f1(h1 \rightarrow s9 \rightarrow s5 \rightarrow s1 \rightarrow s7 \rightarrow s11 \rightarrow h6), f2(h4 \rightarrow s10 \rightarrow s5 \rightarrow s2 \rightarrow s7 \rightarrow s12 \rightarrow h8)$.

Table 2 shows the time, power and energy consumption of the two path planning methods in Fig. 2. When calculating the network energy consumption, we only consider the activated switches. As shown in Table 2, the transmission time of the two flows planned by the link sharing avoidance algorithm is $\frac{m}{c}$, because they do not share links and the bandwidth is c. The transmission time of the two flows planned by the link sharing without avoidance algorithm is $\frac{2*m}{c}$, because they share links, the bandwidth is $\frac{c}{2}$. By calculating the number of activated switches and links, link sharing avoidance algorithm and link share without avoidance algorithm need $\frac{m}{c} * (8a + 24b)$ and $\frac{2*m}{c} * (7a + 20b)$ energy consumption to transmit two flows, respectively. Obviously, link sharing avoidance algorithm consumes less energy.

4.2. Switch aggregation algorithm

According to the analysis of the energy consumption of switch in Section 3, we find that the fixed energy consumption of switch is much greater than that of port, thus the fixed energy consumption of switch is also much larger than the link energy consumption.

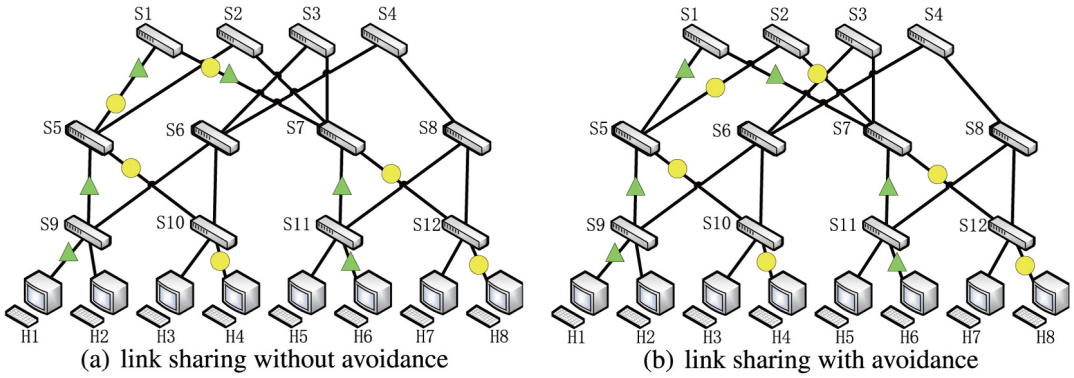


Fig. 2. Comparison of path selection between link sharing without avoidance and link sharing avoidance.

Algorithm 1: Link Sharing Avoidance Algorithm.

Input: weighted network graph G; flag flow state is arriving or finishing *state*;
Output: updated weighted graph G

```
if state == 'arriving' then
    for each i : i ∈ [0, len(path) − 1] do
        src = path[i];
        dst = path[i + 1];
        link = (src, dst);
        G.link.weight = G.link.weight + X_weight;
    end for
end if
if state == 'finishing' then
    for each i : i ∈ [0, len(path) − 1] do
        src = path[i];
        dst = path[i + 1];
        link = (src, dst);
        G.link.weight = G.link.weight − X_weight;
    end for
end if
```

We aggregate the flows to the idle links connected to the activated switches so as to improve the port utilization of activated switches and reduce the number of activated switches.

The main idea of switch aggregation algorithm given in Algorithm 2 is to gather traffics to the idle ports on the activated switch to improve the utilization of switch, by reducing the weight of links connected to the switch. The path set contains all the paths that are being transmitted. First of all, the switch aggregation algorithm obtains all activated switches through the path set. Then, the algorithm determines whether a link in the network is occupied or not. If the link is not occupied and the switches at two ends of the link is activated, the $2a$ power needs to be consumed to activate the link, and the link weight is set to $2{*}P_{weight}$. If the link is not occupied and one of the switches at two ends of the link is activated, the $2a + b$ power needs to be consumed to activate the link, and the link weight is set to $S_{weight}+2{*}P_{weight}$. If the link is not occupied and the switch at two ends of the link is in a state of sleep, $2a{+}2b$ power needs to be consumed to activate the link, and the link weight is set to $2 * S_{weight} + 2 * P_{weight}$.

To illustrate the advantages of switch aggregation algorithm, we give the examples as shown in Fig. 3. We assume that the bandwidth of each link is c, the fixed power of each switch is a and the power of each port is b. Suppose there are two flow requests, $f1(h1 \rightarrow h8)$ and $f2(h2 \rightarrow h6)$. The path planned in the case of no switch aggregation algorithm is: $f1(h1 \rightarrow s9 \rightarrow s5 \rightarrow s1 \rightarrow s7 \rightarrow s12 \rightarrow h8), f2(h4 \rightarrow s10 \rightarrow s6 \rightarrow s3 \rightarrow s7 \rightarrow s11 \rightarrow h6)$. The path planned in the case of using the switch aggregation algorithm is: $f1(h1 \rightarrow s9 \rightarrow s5 \rightarrow s1 \rightarrow s7 \rightarrow s12 \rightarrow h8), f2(h4 \rightarrow s10 \rightarrow s5 \rightarrow s2 \rightarrow s7 \rightarrow s11 \rightarrow h6)$.

Table 3
Comparison of time, power and energy consumption between switch aggregation and non switch aggregation.

	Non switch aggregation	Switch aggregation
Time consumption	$\frac{m}{c}$	$\frac{m}{c}$
Power	$9a + 24b$	$8a + 24b$
Energy consumption	$\frac{m}{c} * (9a + 24b)$	$\frac{m}{c} * (8a + 24b)$

Table 3 compares the time, power and energy consumed by the two flows planned by non switch aggregation algorithm and switch aggregation algorithm in Fig. 3. As shown in Fig. 3, the paths planned by the non switch aggregation algorithm and the switch aggregation algorithm take up the link separately, with bandwidth of c, thus their transmission times are $\frac{m}{c}$. We can find that the switch aggregation algorithm reduces the number of activated switches by aggregating traffic to an already activated switch. By calculating the number of activated switches and links, the non switch aggregation algorithm and the switch aggregation algorithm need to consume the energy of $\frac{m}{c} * (9a + 24b)$ and $\frac{m}{c} * (8a + 24b)$ to transmit two flows, respectively. It is clear that the switch aggregation algorithm consumes less energy than the non switch aggregation algorithm.

4.3. Heuristic algorithm

In SDN, data plane and control plane are separated and communicated by OpenFlow protocol. Once a switch finds a new flow arriving and there is no corresponding matching rule, the switch sends messages to the controller immediately. Subsequently, the controller plans a path for the flow based on the network topology information collected, and then sends the rules to the corresponding switch.

In this subsection, we propose a heuristic algorithm, called Integrated Time and Power (ITP) algorithm as given in Algorithm 3. The main idea of ITP algorithm is to avoid link sharing in time dimension and switch aggregation in power dimension, so as to improve the utilization of switches and achieve the purpose of energy saving in path planning. Specifically, at first, the ITP algorithm detects that whether triggering event is a new flow arriving or the previous flow finishes transmission. If it is a new flow arriving, the Dijkstra algorithm is called to plan a minimum weight path according to the current network state. The weight of the path reflects the energy consumption of the path, so the minimum weight path is the minimum energy consumption path. Then, ITP algorithm adds the path to activate path set, calls link sharing avoidance algorithm, and increases the weight of the occupied link to avoid sharing links with other flows. Furthermore, the switch

Algorithm 2: Switch Aggregation Algorithm.

Input: weighted topology of network G; set of paths for all flows $paths$;
Output: updated weighted graph G

for each $path : path \in paths$ **do**
  for each $i : i \in [0, len(path) - 1]$ **do**
    $node = path[i]$;
    if $node$ not in $AcctiveNodes$ **then**
      $AcctiveNodes = AcctiveNodes + \{node\}$
    end if
  end for
end for
for each $link : link \in G.link$ **do**
  if $link.src \in AcctiveNodes$ and $link.dst \in AcctiveNodes$ **then**
    if $link.weight < X_{weight}$ **then**
      $link.weight = Pweight$
    end if
  end if
  if $link.src \in AcctiveNodes$ or $link.dst \in AcctiveNodes$ **then**
    if $link.weight < X_{weight}$ **then**
      $link.weight = Sweight + 2 * Pweight$
    end if
  end if
  if $link.src \notin AcctiveNodes$ and $link.dst \notin AcctiveNodes$ **then**
    if $link.weight < Xweight$ **then**
      $link.weight = 2 * Sweight + 2 * Pweight$
    end if
  end if
end for

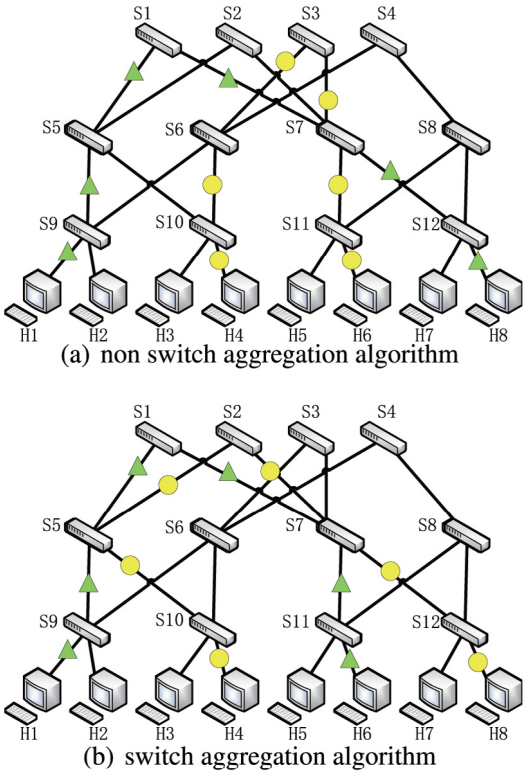


Fig. 3. Comparison of path selection between non switch aggregation and switch aggregation.

aggregation algorithm is called to reduce the weight of idle links on active switches, so that other flows are more willing to pass those links. If a previous flow finishes transmission, then the link sharing avoidance algorithm is called to recover the link's original weight and release the occupied link. Subsequently, the switch aggregation algorithm is used to recalculate activated switches and adjust the link weight according to the new set of activated switches.

Algorithm 3: Integrated time and power algorithm.

Input: weighted topology of network G; a new arriving flow or a finished flow f;
Output: a $path$ for f

$src = f.src$, $dst = f.dst$;
if flow f is a new arriving flow **then**
  $path = dijkstra(G, src, dst)$;
  $paths[(src, dst)] = path$;
  $ResetPathWeight(path, 'arriving')$;
  $ResetGraphWeight(paths, G)$;
end if
if flow f is a finished flow **then**
  $ResetPathWeight(path, 'finishing')$;
  $paths = paths - \{path\}$;
  $ResetGraphWeight(paths, G)$;
end if

Next, we give the complexity analysis of our proposed algorithms. It is assumed that there are l links, n nodes, and m flow requests in the network. It is not difficult to find that, in the worst case, the time complexity of the algorithm is $O(4n + nlog(n) + 3mn + 5l)$ when a new flow arrives, and the time complexity is $O(4n + 3mn + 5l)$ when a flow completes the transmission.

5. Performance evaluation

In this section, we will first evaluate the gap between our proposed heuristic algorithms and the optimal solution (generated by the Optimal Solution Greedy (OSG) algorithm) in a small network. Then we compare the ITP algorithm with BEERS algorithm [16] and the heuristic algorithm proposed in [11], which is called Smallest Closed Sets (SCS) algorithm in this paper. The main purpose of SCS algorithm is to reduce power by aggregating traffic to the minimum set of network elements to achieve saving energy. The BEERS algorithm avoids link sharing by exclusive routing and improves the switch and link utilization by prioritizing the flow with minimum activation time. Finally, we compare these two algorithms in large-scale networks.

5.1. Experiment settings

Operating environment: (1) Hardware configuration: 3.1 GHz CPU, i7-5557U processor, and 4G RAM. (2) Software configuration: We utilize Mininet [17] and Ryu controller [18] to build SDN network environment, and the communication protocol is OpenFlow1.3 [30]. They are implemented in the Ubuntu.04 system.

Network topology: Our topology is as follows: (1) We evaluate the gap between the optimal solution and the heuristic algorithm in the classical Fat-Tree data center topology. Each switch has 4 ports, so the topology contains 20 switches and 16 servers. (2) We evaluate the performance of ITP algorithm and SCS algorithm in three different data center network topologies, i.e., Fat-Tree, Blocking Fat-Tree, and VL2. Three topologies are shown in Fig. 4. (3) In order to evaluate the energy saving performance of ITP algorithm in large-scale networks, we adopt Fat-Tree topology, and each switch has 6 ports, so the topology contains 45 switches and 128 servers.

Traffic generation: In this experiment, traffic requests follow Poisson distribution. The source and destination nodes of each

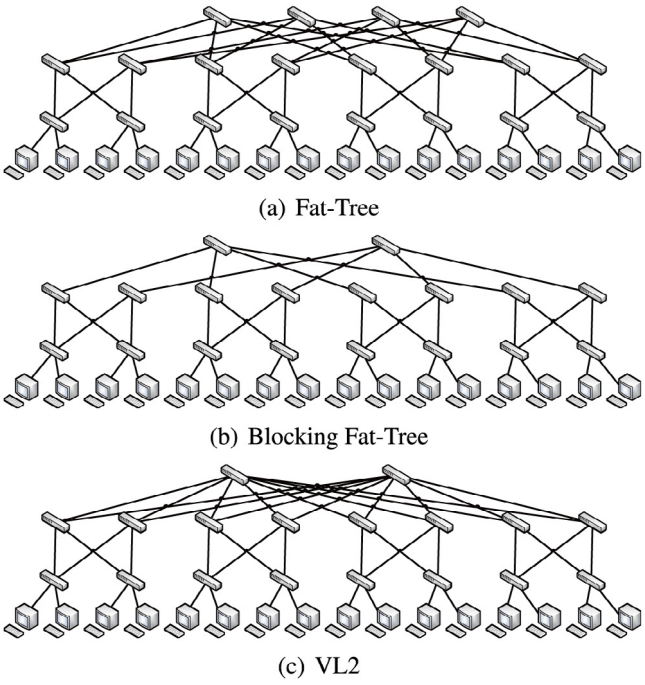


Fig. 4. Topology of Data Center.

request are randomly selected. All flows are generated instantaneously, and the interval between flows is close to zero. Different numbers of flows simulate different network loads. The bandwidth of the flow depends on the bottleneck link in the planned path. The size of all flows is a constant, which is a different value in the following different tests. The service time of a flow depends on the size of the flow, the bandwidth of the flow, and the quality of the links on the planned path. We use iperf [31] to generate real network traffic to evaluate the performance of the algorithm under varying traffic size and quantity.

Number of iterations: Suppose there are n flows in the network. The OSG algorithm explores and compares all possible solutions to find the best solution with an iteration count of $n!$. The ITP algorithm, BEERS algorithm and SCS algorithm are triggered only when a new flow arrives or a previous flow finishes the transmission. Therefore, they are first-come-first-served, and the number of iterations of both algorithms is n.

Energy parameter: We set the parameters used in the experiment as shown in Table 4. Each switch has a fixed power of 48 W, and the power of each switch port is 4 W [32]. The weight proportion coefficient α, β, γ and δ are set to 1, 1, 10 and 20, respectively.

Performance index: We define E_{total} as the network energy consumption without using the energy saving algorithm, E_{actual} represents the actual network energy consumption using the energy saving algorithm, and E_{saved} represents the energy saved by the energy saving algorithm. We define the energy saving percentage, denoted by ESP:

$$ESP = \frac{E_{saved}}{E_{total}} \times 100\% = (1 - \frac{E_{actual}}{E_{total}}) \times 100\%$$

5.2. Comparison of optimal results

We evaluate the gap between the solution of the ITP algorithm and the optimal solution. The optimal solution is generated by the simplest optimal solution greedy (OSG) algorithm. The OSG algorithm explores and compares all possible solutions to find the optimal solution. In the experiment, we compare the computational delay between ITP algorithm and OSG algorithm, and explore the difference between the two algorithms.

Table 4
Parameter setting.

Parameter	Value
$a(u, v)$	4 W
$b(u)$	48 W
α	1
β	1
γ	10
δ	20

As shown in Fig. 5(a), we find that the energy saving performance of the ITP algorithm and the OSG algorithm are the same in most cases, and the maximum gap between the two algorithms is less than 1%. Moreover, the energy saving percentage of the two algorithms are more than 60%. Therefore, the solution of the heuristic ITP algorithm can well approximate the optimal solution of the OSG algorithm, and has a good performance of energy saving. The difference between the ITP algorithm and the OSG algorithm is that the ITP algorithm uses a first-come first-served strategy instead of searching for the optimal scheduling order of the process scheduling.

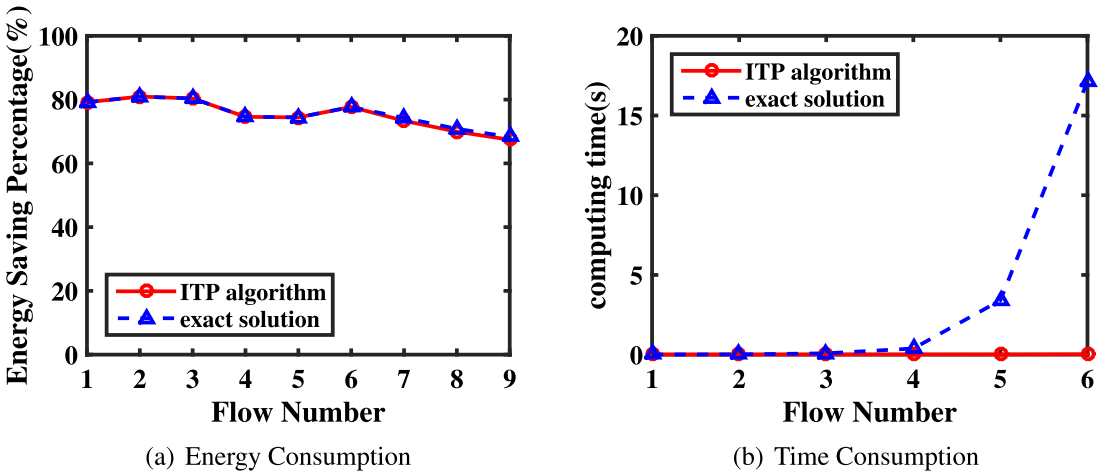
As shown in Fig. 5(b), we find that there is a great gap between the ITP algorithm and the OSG algorithm in terms of computing time. The ITP algorithm needs less than 50 ms when processing 6 flows, while the OSG algorithm needs 17.155 s. This is because the OSG algorithm needs to compare all possible solutions, while ITP uses a first-come, first-served flow processing approach. When the number of flows is 6, the iterations of OSG algorithm and ITP algorithm are 6! and 6 respectively.

5.3. Performance comparison in different network topologies

In order to evaluate the energy saving performance of ITP algorithm in realistic data centers, we use Mininet to create three typical data center topologies, i.e., Fat-Tree, Blocking Fat-Tree and VL2, as shown in Fig. 4. Moreover, we utilize RYU as a controller to monitor the data plane. We use iperf tool [31] to generate traffics with the size of 64M (typical traffic size in a data center) [33]. In the experiment, we compared the transmission time and energy consumed by all flows to complete the transmission using the ITP algorithm, BEERS algorithm and SCS algorithm. Limited by the PC's RAM is only 4G, without loss of generality, we let the number of flow vary from 10 to 50.

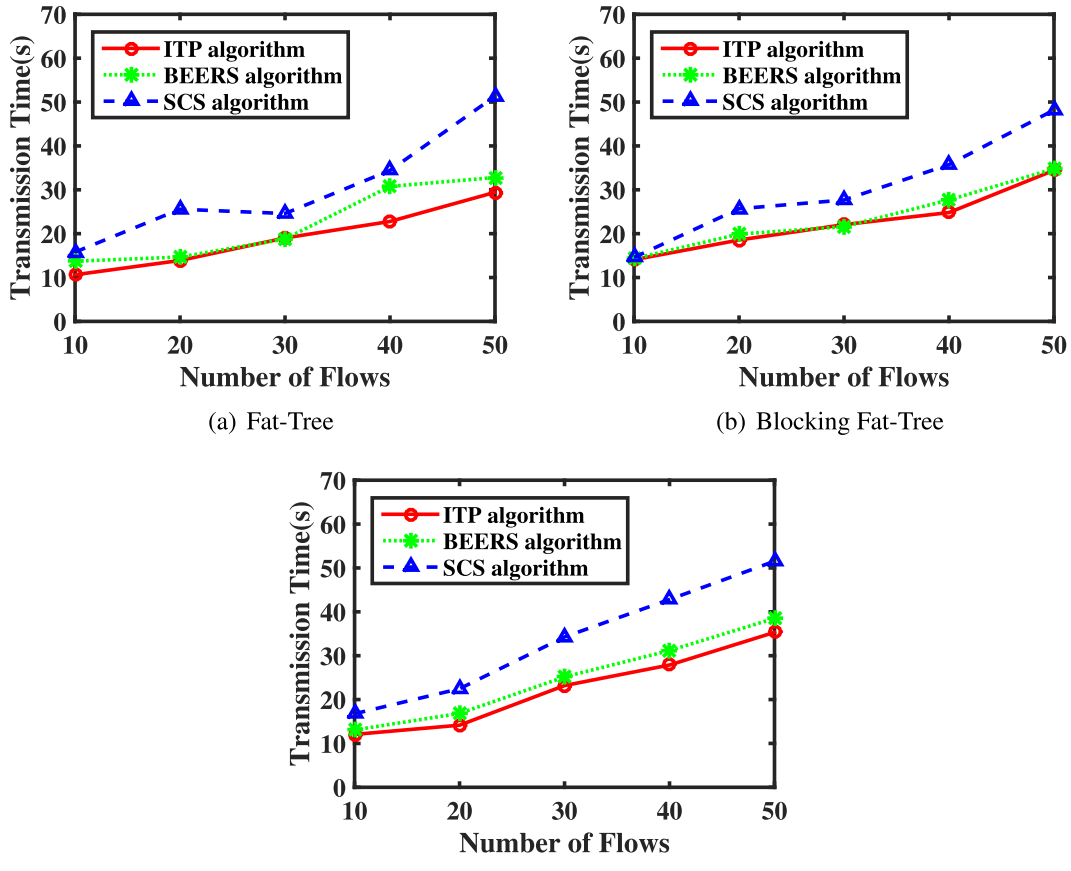
Fig. 6 depicts the performance changes in the completion time of the ITP algorithm, BEERS algorithm and SCS algorithm in the three data center topology, as the number of flows changes. As shown in Fig. 6(a), the transmission time of three algorithms increases when the number of flows increases. The reason under the observation is as follows. For ITP algorithm and SCS algorithm, there exist more flows in the network, resulting in more network bottlenecks and bandwidth reduction. For BEERS, more flows mean more queueing delays. In addition, the transmission time of ITP algorithm is very close to that of BEERS algorithm, and both of them are superior to SCS algorithm. Specifically, in the Fat-Tree network, the ITP algorithm and the BEERS algorithm reduce the transmission time by 36.88% and 26.98% on average compared to the SCS algorithm. This is because ITP algorithm and BEERS algorithm avoid sharing links by link sharing avoidance and exclusive routing respectively, so that the link utilization is improved. Similar results are shown in Fig. 6(b) and (c).

In the Blocking Fat-Tree network, the ITP algorithm and the BEERS algorithm on average reduce the transmission time by 24.99% and 22.14% compared with the SCS algorithm. This means



(a) Energy Consumption (b) Time Consumption

Fig. 5. Comparison of our ITP algorithm and the OSG algorithm.



(a) Fat-Tree (b) Blocking Fat-Tree

(c) VL2

Fig. 6. Total flow completion time of our heuristic ITP algorithm, BEERS algorithm and the SCS algorithm against different flow number in Fat-Tree, Blocking Fat-Tree and VL2 networks.

that the transmission time performance of the ITP algorithm and the BEERS algorithm in the Blocking Fat-Tree network is inferior to that in the Fat-Tree network. This is because the Fat-Tree network has fewer core switches and redundant links, so that fewer alternative paths can be used to avoid sharing links. In the VL2 network, the ITP algorithm and the BEERS algorithm on average reduce the transmission time by 25.68% and 32.91% compared with the SCS algorithm. This result is better than Blocking Fat-Tree, second to Fat-Tree, because VL2 networks have more redundant links than Blocking Fat-Tree networks, but have fewer redundant switches than Fat-Tree networks.

Fig. 7 shows the energy consumption used to transmit all the flows against different number of flows in the Fat-Tree, Blocking Fat-Tree and VL2 networks. In the three kinds of networks, the energy consumed by the ITP algorithm, BEERS algorithm and SCS algorithm increases as the number of flows increases, because more switches need to be activated and their activation time needs to be extended to transmit increased traffic. In Fig. 7(a), we find that the ITP algorithm on average consumes 13.29% less energy

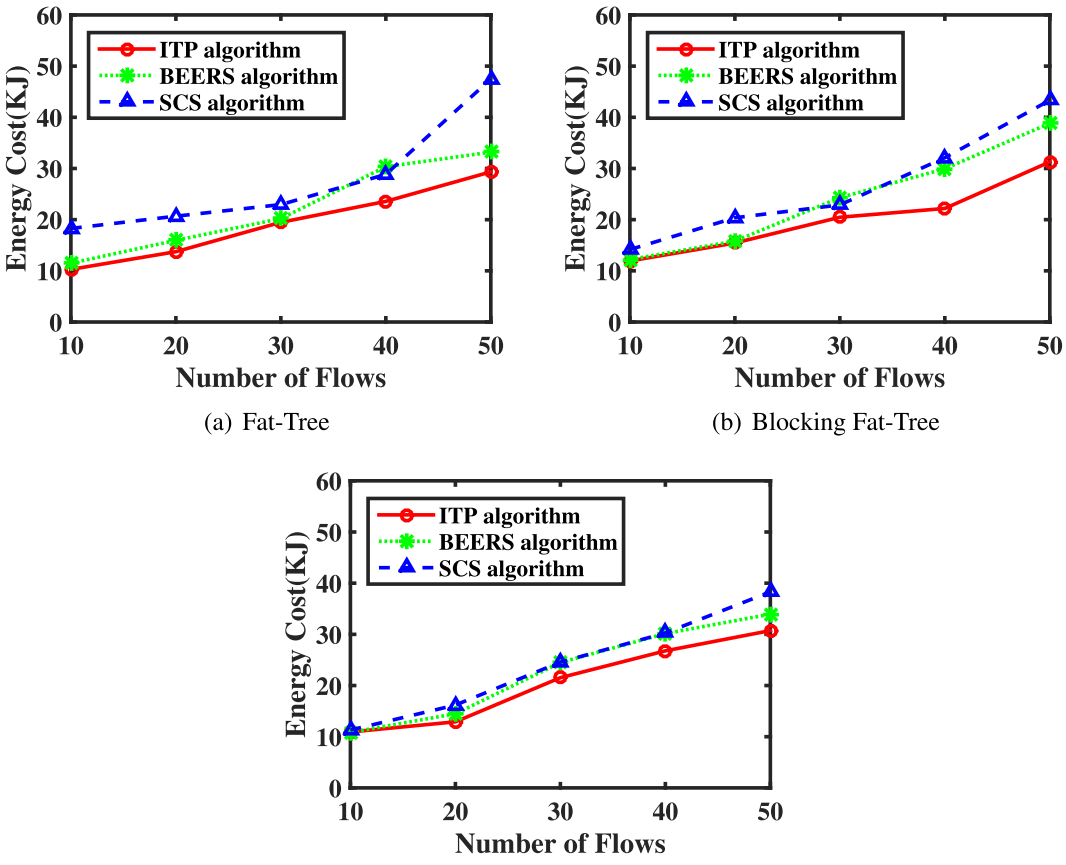


Fig. 7. Energy cost of our heuristic ITP algorithm, BEERS algorithm and the SCS algorithm against different flow number in Fat-Tree,Blocking Fat-Tree and VL2 networks.

than the BEERS algorithm in the Fat-Tree network. The results show that ITP algorithm has better performance of power saving than BEERS algorithm, although they have similar transmission time. Since the ITP algorithm plans a path with lower power for the flow than the BEERS algorithm. Besides, the ITP algorithm on average reduces energy consumption by 30.25% compared to the SCS algorithm. This result demonstrates that the ITP algorithm consumes less energy than the SCS algorithm because the ITP algorithm greatly reduces transmission time by avoiding link sharing, although the ITP algorithm consumes more power than the SCS algorithm.

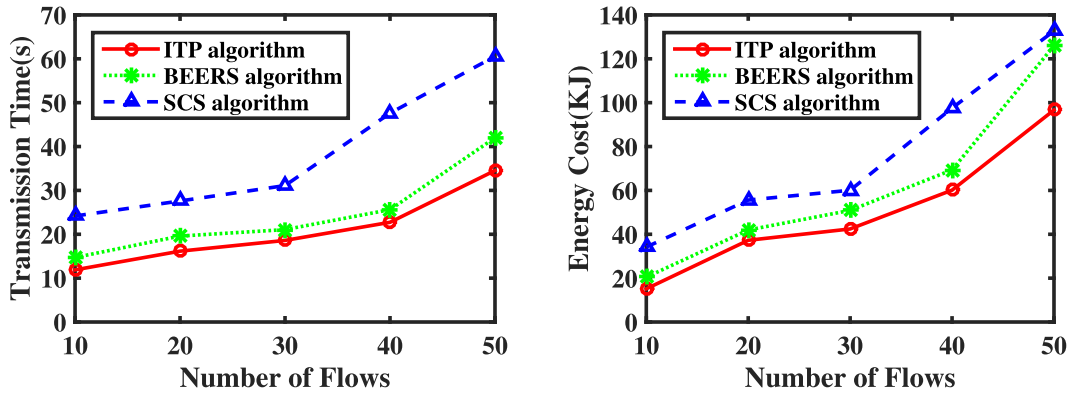
Fig. 7(b) and (c) show the results in Blocking Fat-Tree and VL2 network, with slightly different energy cost due to different topologies. In the Blocking Fat-Tree network, the ITP algorithm reduces energy consumption by an average of 16.32% compared to the BEERS algorithm, which is better than the energy saving in the Fat-Tree network, because fewer switches mean more opportunities for switch aggregation. The ITP algorithm reduces energy consumption by an average of 23.73% compared to the SCS algorithm, which is inferior to the result in Fat-Tree network, because the Blocking Fat-Tree network has fewer redundant links for avoiding sharing. In the VL2 network, the ITP algorithm saves 9.68% and 14.71% of energy, respectively, compared to the BEERS algorithm and the SCS algorithm, which is second to Fat-Tree and Blocking Fat-Tree. Because there are only two core switches in the VL2 network, there are eight links between the core layer and the aggregation layer, which results in most of the energy saved is the link energy consumption. However, the energy consumed by the links is much less than the energy consumed by the switches, occupying a small fraction of the total energy consumption.

5.4. Performance comparison in large-scale networks

In this section, we evaluated the performance of our heuristic ITP algorithm for energy consumption and transmission time in large networks. The network topology we evaluated is a Fat-Tree topology with 6 ports per switch, including 128 servers, and 45 switches. First, we test the performance of the ITP algorithm with a given flow size (64M) and variable number of flows, as shown in Fig. 8. Then, we evaluate the performance of the ITP algorithm for a given number (30 flows) of flows and variable flow size, as shown in Fig. 9.

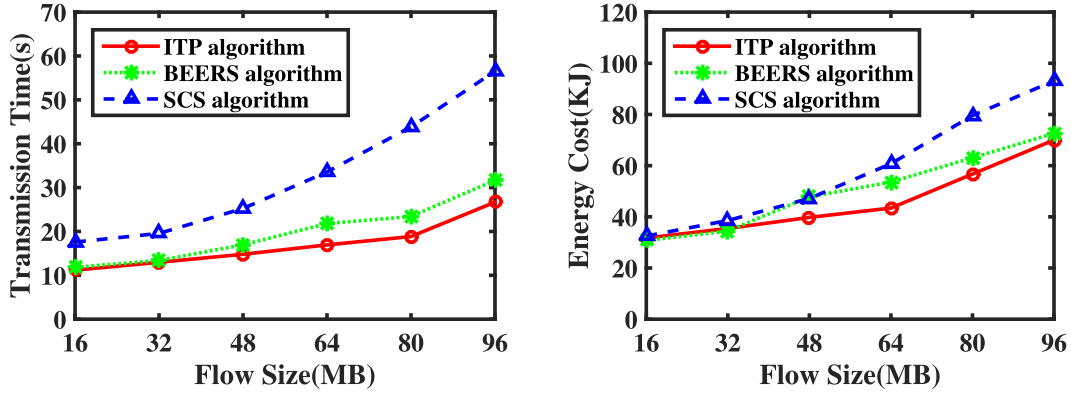
As shown in Fig. 8(a), the transmission time by ITP algorithm, BEERS algorithm and SCS algorithm is naturally prolonged with the increase of the number of flows. In addition, the ITP algorithm and BEERS algorithm are much better than the SCS algorithm in large-scale networks. This is because there are more redundant links in large-scale networks, so that more optional paths can be used to avoid link sharing and exclusive routing. Moreover, we can observe from Fig. 8(b) that with the increase of the number of flows, the energy consumption of ITP algorithm, BEERS algorithm and SCS algorithm significantly increase, owning to more flows need to be transmitted. Furthermore, the ITP algorithm always consumes less energy than the BEERS algorithm and SCS algorithm for different number of flows. As the number of flows increases, the ITP algorithm saves much more energy than the BEERS algorithm and SCS algorithm, because increased flows in the network provide ITP more opportunities to use switch aggregation algorithm and link sharing avoidance algorithm.

As shown in Fig. 9(a), as the size of flows increases, the transmission time of the three algorithms is increased. This is because the increase of flow size prolongs the bottleneck link duration in



(a) Time Consumption (b) Energy Consumption

Fig. 8. Comparison of total flow completion time and energy cost against different flow number in Fat-Tree networks with 128 servers.



(a) Time Consumption (b) Energy Consumption

Fig. 9. Total flow completion time and energy cost against flow size in 128 servers Fat-Tree networks.

SCS algorithm and ITP algorithm, and the queueing time of BEERS algorithm. In addition, ITP algorithm and BEERS algorithm have similar transmission time, while the gap between SCS algorithm and ITP algorithm increases gradually with the increase of flow size. This result demonstrates that link sharing avoidance in the ITP algorithm has a more significant advantage on transmission time when the size of the flow is larger. We can observe from Fig. 9(b) that network devices consume more energy when traffic is larger and more bits need to be transmitted. Besides, the ITP algorithm consumes less energy than the SCS algorithm and BEERS algorithm for different traffic sizes.

6. Conclusions

In this paper, we study the problem of combining the two dimensions of time and power to achieve the purpose of energy saving in data center networks. We first analyze the characteristics of energy consumption in the network and formulate the Minimum Network Energy Consumption (MNEC) problem. Subsequently, the MNEC problem is proved to be a NP-hard problem. Therefore, we design a heuristic algorithm called Integrated Time and Power (ITP) combining link sharing avoidance algorithm and switch aggregation algorithm, which can avoid the link sharing and improve the link utilization from the dimension of time while improving the utilization of the switch port from the dimension of power so as to achieve the purpose of energy saving. Finally, extensive experiments show that ITP heuristic algorithm has better energy saving and transmission delay performance under different network topologies and different network traffic sizes and quantities.

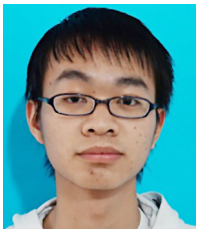
Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61503309, 61772432, 61772433), Natural Science Key Foundation of Chongqing, China (CSTC2016JCYJA0449), Natural Science Foundation of Chongqing, China (cstc2015jcyjBX0094), China Postdoctoral Science Foundation (2016M592619), Chongqing Postdoctoral Science Foundation, China (XM2016002), and the Fundamental Research Funds for the Central Universities, China (XDJK2015C010, XDJK2015D023, XDJK2016A011, XDJK2016D047, XDJK 201710635069).

References

[1] Pierre Delforge, America's data centers consuming and wasting growing amounts of energy, Nat. Resour. Def. Counc. (2014).
[2] Albert Greenberg, James Hamilton, David A. Maltz, Parveen Patel, The cost of a cloud: research problems in data center networks, ACM SIGCOMM Comput. Commun. Rev. 39 (1) (2008) 68–73.
[3] Mohammad Alfares, Alexander Loukissas, Amin Vahdat, A scalable, commodity data center network architecture, in: ACM SIGCOMM Comput. Commun. Rev., vol. 38, ACM, 2008, pp. 63–74.

[4] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, Songwu Lu, BCube:a high performance, server-centric network architecture for modular data centers, ACM SIGCOMM Comput. Commun. Rev. 39 (4) (2009) 63–74.
[5] Dan Li, Chuanxiong Guo, Haitao Wu, Kun Tan, Yongguang Zhang, Songwu Lu, FiConn: Using backup port for server interconnection in data centers, in: INFOCOM, IEEE, 2009, pp. 2276–2285.
[6] Maruti Gupta, Suresh Singh, Greening of the internet, in: Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM, 2003, pp. 19–26.
[7] Maruti Gupta, Satyajit Grover, Suresh Singh, A feasibility study for power management in LAN switches, in: International Conference on Network Protocols, IEEE, 2004, pp. 361–371.
[8] Maruti Gupta, Suresh Singh, Using low-power modes for energy conservation in ethernet LANs, in: IEEE INFOCOM, vol. 7, 2007, pp. 2451–2455.
[9] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, David Wetherall, Reducing network energy consumption via sleeping and rate-adaptation, in: USENIX Symposium on Networked Systems Design and Implementation, vol. 8, 2008, pp. 323–336.
[10] Rui Wang, Zhipeng Jiang, Suixiang Gao, Wenguo Yang, Yinben Xia, Mingming Zhu, Energy-aware routing algorithms in software-defined networks, in: Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, 2014, pp. 1–6.
[11] Huandong Wang, Yong Li, Depeng Jin, Pan Hui, Jie Wu, Saving energy in partially deployed software defined networks, IEEE Trans. Comput. 65 (5) (2016) 1578–1592.
[12] Tingwei Zhu, Dan Feng, Fang Wang, Yu Hua, Qingyu Shi, Yanwen Xie, Yong Wan, A congestion-aware and robust multicast protocol in SDN-based data center networks, J. Netw. Comput. Appl. 95 (2017) 105–117.
[13] Ying-Dar Lin, Yuan-Cheng Lai, Hung-Yi Teng, Chun-Chieh Liao, Yi-Chih Kao, Scalable multicasting with multiple shared trees in software defined networking, J. Netw. Comput. Appl. 78 (2017) 125–133.
[14] Hao Zhu, Xiangke Liao, Cees de Laat, Paola Grosso, Joint flow routing-scheduling for energy efficient software defined data center networks: A prototype of energy-aware network management platform, J. Netw. Comput. Appl. 63 (2016) 110–124.
[15] Dan Li, Yunfei Shang, Wu He, Congjie Chen, EXR: Greening data center network with software defined exclusive routing, IEEE Trans. Comput. 64 (9) (2015) 2534–2544.
[16] Guan Xu, Bin Dai, Benxiong Huang, Jun Yang, Sheng Wen, Bandwidth-aware energy efficient flow scheduling with SDN in data center networks, Future Gener. Comput. Syst. 68 (2017) 163–174.
[17] Mininet, http://mininet.org/. (Online; Accessed 24 December 2017).
[18] Ryu, http://osrg.github.io/ryu/. (Online; Accessed 24 December 2017).
[19] Maruti Gupta, Suresh Singh, Dynamic ethernet link shutdown for energy conservation on ethernet links, in: Communications, 2007. ICC'07. IEEE International Conference on, IEEE, 2007, pp. 6156–6161.
[20] Tao Chen, Xiaofeng Gao, Guihai Chen, The features, hardware, and architectures of data center networks: A survey, J. Parallel Distrib. Comput. 96 (2016) 45–74.
[21] Joseph Chabarek, Joel Sommers, Paul Barford, Cristian Estan, David Tsiang, Steve Wright, Power awareness in network design and routing, in: INFOCOM 2008. The 27th Conference on Computer Communications. IEEE, IEEE, 2008, pp. 457–465.
[22] Brandon Heller, Srinivasan Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, Nick McKeown, ElasticTree: Saving energy in data center networks, in: Usenix Conference on Networked Systems Design and Implementation, vol. 10, 2010, pp. 249–264.
[23] Mingui Zhang, Cheng Yi, Bin Liu, Beichuan Zhang, GreenTE: Power-aware traffic engineering, in: The IEEE International Conference on Network Protocols, IEEE, 2010, pp. 21–30.
[24] Yujie Liu, Yong Li, Yue Wang, Jian Yuan, Optimal scheduling for multi-flow update in software-defined networks, J. Netw. Comput. Appl. 54 (2015) 11–19.
[25] Jia Zhao, Jiangchuan Liu, Haiyang Wang, Chi Xu, Multipath TCP for datacenters: From energy efficiency perspective, in: IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, 2017, pp. 1–9.
[26] Priya Mahadevan, Puneet Sharma, Sujata Banerjee, Parthasarathy Ranganathan, A power benchmarking framework for network devices, in: International Ifip-Tc 6 NETWORKING Conference, Springer, 2009, pp. 795–808.
[27] Amitabha Banerjee, Wu Chun Feng, Biswanath Mukherjee, Dipak Ghosal, RAPID: an end-system aware protocol for intelligent data transfer over lambda grids, in: Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. International, 2006, p. 10.
[28] UDT (UDP-based Data Transfer), http://udt.sourceforge.net. (Online; Accessed 24 December 2017).
[29] Qishi Wu, N.S.V. Rao, A class of reliable UDP-based transport protocols based on stochastic approximation, in: INFOCOM 2005. Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, vol. 2, 2005, pp. 1013–1024.
[30] OpenFlow Switch Specification Version 1.3, Open Network Foundation.
[31] Cisco Nexus 2200 series data sheet, https://github.com/esnet/iperf. (Online; Accessed 24 December 2017).
[32] Cisco Nexus 2200 series data sheet, http://www.cisco.com/en/US/prod/collateral/switches/ps9441/. (Online; Accessed 24 December 2017).
[33] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung, The Google file system, in: Nineteenth ACM Symposium on Operating Systems Principles, vol. 37, ACM, 2003, pp. 29–43.



Yue Zeng received the B.S. degree in Computer science and Engineering from Chongqing Three Gorges University, Chongqing, China, in 2016. He is currently working toward the M.S. degree in signal and information processing, Southwest University. His research interests include network energy saving and software defined networking.

Songtao Guo received the BS, MS, and PhD degrees in computer software and theory from Chongqing University, Chongqing, China, in 1999, 2003, and 2008, respectively. He was a professor from 2011 to 2012 at Chongqing University. He is currently a full professor at Southwest University China. He was a senior research associate at the City University of Hong Kong from 2010 to 2011, and a visiting scholar at Stony Brook University, New York, from May 2011 to May 2012. His research interests include wireless networks, mobile cloud computing and parallel and distributed computing. He has published more than 80 scientific papers in leading refereed journals and conferences. He has received many research grants as a principal investigator from the National Science Foundation of China and Chongqing and the Postdoctoral Science Foundation of China.

Guiyan Liu received the B.S. degree in telecommunications engineering from Southwest University, Chongqing, China, in 2014. She is currently working toward the Ph.D. degree in signal and information processing, Southwest University. Her research interests include stream scheduling in data center networks and software defined networking.