Energy-Efficient Device Activation, Rule Installation and Data Transmission in Software Defined DCNs

Yue Zeng, Songtao Guo, *Senior Member, IEEE*, Guiyan Liu, Pan Li, and Yuanyuan Yang, *Fellow, IEEE*

Abstract—With the prosperity of cloud computing and video services, the demand for network resources has increased dramatically, leading to the remarkable growth in the amount of network energy consumption, a key factor restricting the development of data centers. Numerous existing works reduce network energy consumption by optimizing data transmission, but they ignore the energy consumption for data transmission preparation, such as activating devices and installing rules. In this paper, we jointly optimize device activation, rule installation and data transmission to minimize network energy consumption. Specifically, we first formulate the minimization problem of the energy consumption of device activation, rule installation, and data transmission. We then prove that it is NP-complete to get the optimal solution of the minimization problem, furthermore, we propose a heuristic algorithm to plan the path with minimum network energy consumption for each flow. The simulation results show that the energy consumption of our algorithm is close to the optimal solution solved by Gurobi, and our algorithm has lower complexity. Compared with the state-of-the-art algorithm, our algorithm always consumes less energy and has shorter flow completion time.

Index Terms—Device activation, rule installation, data transmission, energy minimization management, software defined data center networks

1 Introduction

With the exponential growth in demand for cloud computing and video services, the size and number of data centers have exploded, leading to a dramatic increase in energy consumption. It is reported that in 2013, the energy consumption of data centers in the United States has reached 91 billion kilowatt-hours, and will increase to 140 billion kilowatt-hours by 2020 [1]. Furthermore, the ratio of network energy consumption over the entire data center energy consumption will increase to 50 percent in the future [2]. Therefore, network energy conservation in the data center has become a hot topic [3], [4].

Data center networks are designed to meet peak traffic demands, however, network load is not always at its peak. Mostly, the data center network is under low load or idle state, and even the core layer's link utilization is less than 25 percent [5]. This means that low-load or idle network devices consume most of the network energy. An intuitive way to save energy is to set those idle network devices to sleep. In traditional data centers, several energy-saving strategies have been proposed, such as shutting down idle Ethernet links [6], adjusting the transmission rate of devices to adapt to network loads [7], using low-power modes for switches [8]. However, in traditional data center networks, switches can only get information from surrounding nodes, and the global optimization of network energy consumption cannot be achieved.

With the emergence of Software Defined Network (SDN) technology, network management has a global network perspective. Therefore, much literature has studied the network energy conservation from a global perspective and proposed a list of network energy conservation strategies [3], [9], [10], [11], [12], [13], [14]. On the one hand, several existing efforts [9], [10], [11], [12] focus on reducing network power by aggregating traffic to the smallest set of network elements and setting unnecessary devices to sleep. On the other hand, the exclusive routing method is used to improve the link utilization rate and reduce the data transmission delay, thereby reducing the energy consumption of data transmission [13], [14]. Furthermore, the FCTcon algorithm [11] reduces flow completion time and network power by traffic aggregation and dynamic bandwidth control. The above methods effectively save data transmission energy consumption in the network.

In energy-efficient software defined networks, before transmitting a flow, one needs to activate all sleeping network devices on the routing path, and ensure that all switches on the routing path have forwarding rules installed. Thus, the

- *Y. Zeng, G. Liu, and P. Li are with in the Chongqing Key Lab of Nonlinear Circuits and Intelligent Information Processing, College of Electronic and Information Engineering, Southwest University, Chongqing 400715, P. R. China. E-mail: {1477955264, guiyanliu0102, 942907022}@qq.com.*
- *S. Guo is with the College of Computer Science, Chongqing University, Chongqing 400044, P. R. China, and also with Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, China, and also with the College of Electronic and Information Engineering, Southwest University, Chongqing 400715, P. R. China. E-mail: songtao_guo@163.com.*
- *Y. Yang is with the Department of Electrical & Computer Engineering, Stony Brook University, Stony Brook, NY 11794 USA. E-mail: yuanyuan.yang@stonybrook.edu.*

Manuscript received 6 May 2019; revised 14 Sept. 2019; accepted 12 Oct. 2019. Date of publication 16 Oct. 2019; date of current version 8 Mar. 2022. (Corresponding author: Songtao Guo.)
Recommended for acceptance by C. Wu.
Digital Object Identifier no. 10.1109/TCC.2019.2947900

flow energy consumption consists of device activation energy consumption, rule installation energy consumption and data transmission energy consumption. Rule installation energy consumption is not negligible for short flow while the device activation energy consumption is not negligible for both short flow and long flow (detail in the Section 3). At present, the study for device activation and rule installation aims to reduce the delay. However, how to optimize these two kinds of energy consumption has not been considered in the existing works, resulting in a sharp decline in the performance of network energy saving.

In this paper, we jointly optimize the energy consumption of device activation, rule installation and data transmission. Specifically, we propose a Green Network (GN) algorithm, which handles long flows and short flows separately, according to their energy consumption characteristics. On the one hand, for a long flow, device activation energy consumption is not negligible. The GN algorithm reduces device activation energy consumption by avoiding routing a long flow to sleep switches. On the other hand, for a short flow, both device activation energy consumption and rule installation energy consumption cannot be ignored. The GN algorithm preferentially forwards a short flow using the installed forwarding rules and activated devices in the network, which can effectively reduce the device activation energy consumption and the rule installation energy consumption. Furthermore, the GN algorithm constructs a weighted graph for each flow, and the weight of each link reflects the energy consumption of the flow. In other words, the more energy a flow consumes on a link, the greater the weight of the link. Therefore, we transform the energy-minimum path planning problem into the minimum weighted path search problem, which greatly reduces the complexity of the problem.

The major contributions of this paper can be summarized as follows:

- We build the Joint energy consumption Minimization problem of Device activation, Rule installation and Data transmission (JMDRD), and prove that the JMDRD problem is NP-complete.
- We propose an efficient heuristic algorithm that takes into account device activation, rule installation, and data transmission to plan an energy-saving path for each flow.
- We conduct extensive experiments using a real Internet traffic matrices trace [15]. The experimental results show that our algorithm is close to the optimal solution in energy saving. Compared with the existing algorithms, our algorithm has shorter Flow Completion Time (FCT) and consumes less energy.

The rest of paper is organized as follows: Section 2 introduces the related work. Section 3 provides two examples to illustrate the motivation of this paper. In Section 4, we build the JMDRD model and prove that it is NP-complete. Then, we propose a heuristic algorithm in Section 5 and evaluate it in Section 6. Finally, we conclude this paper in Section 7.

2 Related Work

In this section, we first introduce energy saving efforts in traditional data center networks and software defined networks. After that, we introduce the works related with rule installation and device activation.

In traditional networks, all network elements are independent of each other and can only obtain information from the surrounding nodes. Therefore, most of the existing works focused on local network energy conservation [6], [7], [8], [9], [11], [16]. First, Gupta et al. [16] studied the possibility of setting various components on a LAN switches to sleep during low traffic load. Then, they proposed saving energy by dynamically shutting down idle Ethernet links [6] or using low-power modes when the Ethernet transceiver is underutilized [8]. Furthermore, two power management schemes [7] were presented to reduce network energy consumption, such as putting idle network components to sleep, adapting the network operation rate to the workload. However, local network energy saving is far from the global optimum.

The emergence of software defined network technology makes it possible to manage network energy from a global perspective. Then, some works studied the network energy saving from a global perspective [3], [9], [10], [11], [12], [13], [14], [17]. On the one hand, some existing research works focused on optimizing the data transmission power in the network [9], [10], [12], [17]. Wang et al. [17] proposed a 0-1 ILP model to minimize the power of the integrated chassis and line-cards. However, solving ILP is time-consuming. Then, ElasticTree algorithm for tree networks [9] and heuristic algorithm for partially deployed software defined networks [10] were proposed, which aggregate flows into the smallest set of network devices and shut down idle devices to save network energy. In addition, [12] minimized the data transmission power of heterogeneous wireless cloud access networks by minimizing the number of active small base stations.

On the other hand, some research works optimized the data transmission delay in the network [13], [14]. Li et al. [13] explored a new solution to energy-aware flow scheduling and proposed an EXR scheduling scheme, i.e., a flow always exclusively occupies the link on its routing path. On this basis, [14] proposed a BEERS algorithm, which further satisfies the traffic requirements such as deadline. The ITP algorithm [3] and the FCTcon algorithm [11] combined the power and time of data transmission to optimize the network energy consumption.

There are also some works to optimize rule installation in software defined networks. Reitblatt et al. [18] proposed a two-phase commit protocol to avoid black holes in the rule installation process, but the protocol doubles the precious memory overhead. Then, Jedidiah et al. [19] presented an order-based algorithm that guarantees no black holes without sacrificing memory, which prolongs the rule installation delay. FLIP [20] combines the advantage of both two-phase commit protocol and order-based algorithm, which significantly reduces memory overhead and rule installation latency. However, the above methods do not explore how to optimize rule installation to minimize network energy consumption. In addition, [13] recognized the impact of device activation on energy consumption performance and tested their algorithms at different device activation duration settings. However, they neither built a model nor proposed an algorithm to optimize device activation energy consumption.

Different from the existing work, in this paper we jointly optimize the energy consumption of device activation, rule

installation and data transmission to minimize network energy consumption.

3 Motivation

In this section, we first provide the components of flow energy consumption by analyzing the process of a flow in SDNs. Then, we give two examples to illustrate the motivation of this paper.

3.1 Components of Flow Energy Consumption

In SDNs, when a new flow arrives at a switch, the switch looks for the forwarding rule of that flow from its flow table. If the forwarding rule is found, the flow is forwarded. Otherwise, the switch notifies the controller through the OpenFlow protocol [21], and then the controller assigns a path to the flow. In energy-efficient networks, idle network devices need to be set to sleep for energy saving. Once the path planned for a flow contains sleeping network devices, the controller must activate those devices. Then, the controller sends the forwarding rules to the data plane, and the switches on the routing path install the forwarding rules, and the flow is forwarded.

In summary, the flow processing mainly consists of three phases: device activation, rule installation, and data transmission. The device activation phase includes link activation and switch activation. Generally, it takes about 10 ms to activate a link [13], which is obviously not negligible for short flow. It takes a few seconds to activate a switch [7], [13], [22], which is not negligible for both long and short flow. It takes about 10 ms for a switch to install a forwarding rule from the controller [23], [24], which is also not negligible for short flow. The data transmission phase is an important part of flow processing. Therefore the energy consumption of a flow can be divided into device activation energy consumption, rule installation energy consumption, and data transmission energy consumption.

In the following, we will give two examples to show that device activation energy consumption and rule installation energy consumption play an important role in energy saving, and optimization of both can significantly promote network energy saving.

3.2 Device Activation Energy Consumption

The transmission of a flow needs to pass through switches and links. Therefore, the device activation energy consumption can be divided into link activation energy consumption and switch activation energy consumption. Frequent activation of network devices results in a dramatic increase in network energy consumption and a decline in quality of service, such as extending the FCT.

Here, we propose an Optimizing Device Activation Energy Consumption (ODAEC) algorithm and a Non-Optimizing Device Activation Energy Consumption (Non-ODAEC) algorithm. The ODAEC algorithm optimizes both device activation energy consumption and data transmission energy consumption, while Non-ODAEC algorithm only optimizes data transmission energy consumption. The two algorithms are compared in Fig. 1 to highlight the importance of optimizing device activation energy consumption.

As shown in Fig. 1, there is a flow $f_m = \{h_1, h_2, c_m, r_m\}$ in the network, where $\{h_1, h_2, c_m, r_m\}$ represents the source, destination, bandwidth and size of flow f_m, respectively. In this example, each link in the network can satisfy the bandwidth request of f_m, that is, the capacity of each link is greater than or equal to c_m. The time required to activate a link is ς. The power of each link and each switch is a and b, respectively. The data transmission duration of flow f_m is equal to the time required to activate a link, that is, $\frac{r_m}{c_m} = \varsigma$. In order to simplify the analysis of device activation energy consumption, we assume that the forwarding rule of flow f_m has been installed, and no rule installation energy consumption is required.

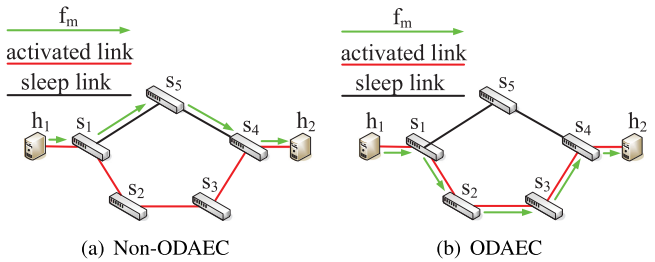


Fig. 1. Comparison of the Non-ODAEC algorithm with the ODAEC algorithm to highlight the importance of optimizing device activation. In the network, all switches and partial links $(h_1 - s_1, s_1 - s_2, s_2 - s_3, s_3 - s_4, s_4 - h_2)$ have been activated and other links $(s_1 - s_5, s_5 - s_4)$ are in sleep state. When a flow f_m arrives, the path that the Non-ODAEC algorithm plans is $p_1 = (h_1 - s_1 - s_5 - s_4 - h_2)$ for the flow, while the path that the ODAEC algorithm plans is $p_2 = (h_1 - s_1 - s_2 - s_3 - s_4 - h_2)$.

It is not difficult to observe from Fig. 1a that the path planned by Non-ODAEC algorithm for flow f_m is $h_1 - s_1 - s_5 - s_4 - h_2$, which has the minimum data transmission energy consumption. However, the path contains sleep links, which require additional time and energy to activate. Those idle devices can be set to sleep. We can obtain that the energy consumption and FCT of flow f_m are $(\varsigma + \frac{r_m}{c_m}) \times (4a + 3b)$ and $\varsigma + \frac{r_m}{c_m}$, respectively. Fig. 1b shows that the path planned by ODAEC algorithm for flow f_m is $h_1 - s_1 - s_2 - s_3 - s_4 - h_2$, because it consumes less total energy. The network devices on this path have been activated, and the flow can be forwarded directly without device activation energy consumption. By calculation, we can obtain that the energy consumption and FCT of flow f_m are $\frac{r_m}{c_m} \times (5a + 4b)$ and $\frac{r_m}{c_m}$, respectively.

According to the above analysis, we can find that compared with the Non-ODAEC algorithm, the ODAEC algorithm can reduce the energy consumption of switches by 33.33 percent, the energy consumption of links by 37.5 percent and FCT by 50 percent. Then, we can conclude that for a flow whose data transmission duration is close to or less than the link activation duration, i.e., short flow, its energy consumption can be significantly reduced by optimizing the device activation energy consumption. Similarly, for a flow with data transmission duration close to or less than the switch activation duration, i.e., long and short flow, their energy consumption can also be significantly reduced by optimizing device activation energy consumption.

3.3 Rule Installation Energy Consumption

In SDNs, frequent modification of forwarding rules leads to an increase in rule installation energy consumption and an extension of FCT. Rule installation energy consumption can

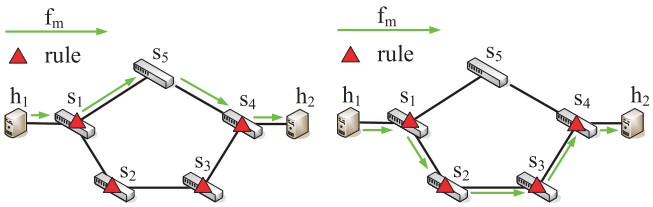


Fig. 2. Comparison of the Non-ORIEC algorithm with the ORIEC algorithm to highlight the importance of optimizing rule installation. In the network, all network devices have been activated and the forwarding rules of flow f_m are installed on partial switches (s_1, s_2, s_3, s_4). When a flow f_m arrives, the path that the Non-ORIEC algorithm plans is $p_3 = (h_1 - s_1 - s_5 - s_4 - h_2)$ for the flow, while the path that the ORIEC algorithm plans is $p_4 = (h_1 - s_1 - s_2 - s_3 - s_4 - h_2)$.

be effectively reduced by utilizing the existing forwarding rules to forward flows.

In this part, we propose an Optimizing Rule Installation Energy Consumption (ORIEC) algorithm and a Non-Optimizing Rule Installation Energy Consumption (Non-ORIEC) algorithm. The ORIEC algorithm optimizes both data transmission energy consumption and rule installation energy consumption, while the Non-ORIEC algorithm only optimizes data transmission energy consumption. The two algorithms are compared to demonstrate the advantages of optimizing rule installation energy consumption.

As shown in Fig. 2, there is also a flow $f_m = \{h_1, h_2, c_m, r_m\}$ in the network. In this example, the time required to install the rule is τ, and the rule installation duration is equal to the data transmission duration of the flow, that is, $\tau = \frac{r_m}{c_m}$.

Fig. 2a shows that the Non-ORIEC algorithm routes the flow f_m to $h_1 - s_1 - s_5 - s_4 - h_2$, with minimal data transmission energy consumption. However, rule installation energy consumption is required to install new forwarding rules. By calculation, we can get that the FCT and network energy consumption of f_m are $\tau + \frac{r_m}{c_m}$ and $(\tau + \frac{r_m}{c_m}) \times (3a + 4b)$, respectively. Fig. 2b shows that the path planned by the ORIEC algorithm for flow f_m is $h_1 - s_1 - s_2 - s_3 - s_4 - h_2$, which does not need to modify the forwarding path. By calculation, we can get that the FCT of the flow is τ, and the energy consumed by the flow is $\tau \times (4a + 5b)$.

Similarly, we can conclude that: for a short flow whose data transmission duration is less than or equal to the rule installation duration, their energy consumption and FCT can be effectively reduced by optimizing the rule installation energy consumption.

4 Problem Formulation

In this section, we first analyze the flow energy consumption and network energy consumption. Subsequently, we build a Joint energy consumption minimization problem of Device activation, Rule installation, and Data transmission (JMDRD) problem. Finally, we prove that the JMDRD problem is NP-complete.

4.1 Analysis of Flow Energy Consumption

In the network, the flow forwarding needs to pass through switches and links. The energy consumed by a flow on each network device depends on the extended duration on the device and the power of the device. Thus, the network energy consumption of a flow m can be expressed as follows:

$$E^m = \sum_{u,v \in L} a_{u,v} \times T^m_{u,v} + \sum_{s \in S} b_s \times T^m_s, \tag{1}$$

where $T^m_{u,v}$ and T^m_s respectively indicate the extended duration of link (u, v) and switch s for transmitting flow m. $a_{u,v}$ and b_s represent the power of link (u, v) and switch s. The notions to be used in this paper are summarized in Table 1.

If a flow passes through a link (u, v) and its duration is shorter than the duration of flow m, then the duration of the link is extended by the flow. Otherwise, the duration of link (u, v) is not prolonged by flow m. The extended duration of link (u, v) by flow m can be expressed as

$$T^m_{u,v} = \begin{cases} T^m - T^{pre}_{u,v}(m), & \text{if } T^m \times X^m_{u,v} > T^{pre}_{u,v}(m), \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

TABLE 1
List of Notations

Symbol	Notations
S	Set of switches
L	Set of links
M	Number of flows
E	Network energy consumption
E^m	Energy consumption of flow m
T^m	The duration of flow m
$T_{u,v}$	The duration of link (u, v)
T_s	The duration of switch s
$T^m_{u,v}$	The extended duration of link (u, v) by flow m
T^m_s	The extended duration of switch s by flow m
$T^{pre}_{u,v}(m)$	The duration of link (u, v) before flow m is scheduled
$T^{pre}_s(m)$	The duration of switch s before flow m is scheduled
T^m_{trans}	Data transmission duration of flow m
T^m_{active}	Device activation duration of flow m
T^m_{rule}	Rule installation duration of flow m
X^m_s	Binary variable indicates whether flow m passes through switch u
$X^m_{u,v}$	Binary variable indicates whether flow m passes through link (u, v)
Y^m_s	Binary variable indicates whether flow m is the longest duration flow passes switch s
$Y^m_{u,v}$	Binary variable indicates whether flow m is the longest duration flow passes link (u, v)
$A_{u,v}$	Binary constant indicate whether link (u, v) is asleep
A_s	Binary constant indicate whether switch s is asleep
$f^m_{u,v}$	The traffic of flow m on link (u, v)
(u, v)	Link from node u to node v
$a_{u,v}$	The power of link (u, v)
b_s	The power of switch s
$c_{u,v}$	Bandwidth capacity of link (u, v)
r^m	Data size of flow m
κ^m	Required bandwidth of flow m
α^m	Binary variable indicates whether sleep switches are included on the routing path of flow m
β^m	Binary variable indicates whether sleep links are included on the routing path of flow m
γ^m	Binary variables indicate whether a new path is assigned to flow m
σ	Constant indicates switch activation duration
ς	Constant indicates link activation duration
τ	Constant indicates the rule installation duration

where $X^m_{u,v}$ indicates whether flow m passes through the link (u, v), T^m denotes the duration of flow m, and $T^{pre}_{u,v}(m)$ indicates the duration of link (u, v) before flow m is scheduled.

$T^{pre}_{u,v}(m)$ is equal to the duration of the flow with the longest duration on link (u, v) before flow m is scheduled, which can be expressed as

$$T^{pre}_{u,v}(m) = \max\{T^j * X^j_{u,v}\}, \forall j \prec m. \tag{3}$$

Similarly, T^m_s and $T^{pre}_s(m)$ can be expressed as can be expressed as

$$T^m_s = \begin{cases} T^m - T^{pre}_s(m), & \text{if } T^m \times X^m_s > T^{pre}_s(m), \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

$$T^{pre}_s(m) = \max\{T^j * X^j_s\}, \forall j \prec m, \tag{5}$$

where $T^{pre}_s(m)$ indicates the duration of switch s before flow m is scheduled.

According to the analysis in Section 3, the flow processing can be divided into device activation phase, rule installation phase and data transmission phase. Therefore, the duration of flow can be expressed as

$$T^m = T^m_{active} + T^m_{rule} + T^m_{trans}, \tag{6}$$

where T^m_{active}, T^m_{rule} and T^m_{trans} denote the device activation duration, rule installation duration and data transmission duration of flow m, respectively.

The data transmission duration for a flow represents the time it takes to transmit the flow, which occupies the major part of the total duration of the flow. In the data plane, we use fixed bandwidth protocols to transport flows, such as FRTP [25], Tsunami [26] and PA-UDP [27]. These protocols allocate a fixed bandwidth for each flow on its routing path, and each port provides a fixed rate for each flow. These flows are independent of each other. Once a flow is allocated the required bandwidth, the flow can be transmitted immediately. Therefore, the data transmission duration of a flow is equal to its size divided by its bandwidth. The data transmission duration of flow m can be expressed as

$$T^m_{trans} = \frac{r^m}{\kappa^m}, \tag{7}$$

where r^m and κ^m denote the data size and the bandwidth required by flow m, respectively.

Device activation duration indicates the time required to activate those sleep network devices, including notification duration, switch activation duration and link activation duration. As mentioned in Section 3, the link activation duration and the switch activation duration cannot be ignored. In data center, sending a message from the controller to the switches causes a delay of less than 0.01 ms [23], [24], [28], which is much less than the link activation duration ($\sim$10 ms [7], [13], [22]), switch activation duration (several second [7], [13], [22]) and rule installation duration ($\sim$10 ms [24]). Thus, the notification duration is negligible. Here, we denote the link activation duration as ς and the switch activation duration as σ. Then, the device activation duration of flow m can be expressed as

$$T^m_{active} = \sigma \times \alpha^m + \varsigma \times \beta^m, \tag{8}$$

where α^m and β^m are binary variables, indicating whether flow m needs to activate links and switches, respectively.

The rule installation duration is the time it takes for switches to install the rules issued from the controller. If the forwarding rule for a flow already exists in the network, then its forwarding path does not need to be modified and its rule installation duration is 0. Otherwise, the flow takes time to install the rules. We assume that the rule installation duration of a flow is a constant τ. Then, the rule installation duration of flow m can be expressed as

$$T^m_{rule} = \tau \times \gamma^m, \tag{9}$$

where T^m_{rule} represents the time that flow m takes to install rules, and γ^m is a binary variable indicating whether flow m needs to install a rule.

According to the formula ((6), (7), (8), (9)), the duration of flow m can be expressed as

$$T^m = \sigma \times \alpha^m + \varsigma \times \beta^m + \tau \times \gamma^m + \frac{r^m}{\kappa^m}. \tag{10}$$

4.2 Analysis of Network Energy Consumption

Network energy consumption is the amount of energy consumed for transmitting all flows in the network, which is also equal to the energy consumed by all network devices. Therefore, network energy consumption can be expressed as

$$\begin{aligned} E &= \sum_{m=1}^{M} E^m \\ &= \sum_{u,v \in L} a_{u,v} \times T_{u,v} + \sum_{s \in S} b_s \times T_s, \end{aligned} \tag{11}$$

where E represents the network energy consumption.

The duration of a switch is equal to the duration of the longest duration flow passing through the switch. Thus, the duration of switch s can be expressed as

$$T_s = \sum_{m=1}^{M} T^m \times Y^m_s, \tag{12}$$

where Y^m_s is a binary variable indicating whether flow m is the longest duration flow that passes through switch s.

Similarly, the duration of the link (u, v) can be expressed as

$$T_{u,v} = \sum_{m=1}^{M} T^m \times Y^m_{u,v}, \tag{13}$$

where $Y^m_{u,v}$ indicates whether flow m is the longest duration flow passing through link (u, v).

According to Equations (11)(12)(13), network energy consumption can be rewritten as

$$E = \sum_{u,v \in L} \sum_{m=1}^{M} a_{u,v} \times T^m \times Y^m_{u,v} + \sum_{s \in S} \sum_{m=1}^{M} b_s \times T^m \times Y^m_s. \tag{14}$$

4.3 Problem Formulation

In this paper, we study the problem of Joint Minimization of Device activation energy consumption, Rule installation energy consumption, and Data transmission energy

consumption (JMDRD). The objective function is set to

$$\min \sum_{u,v\in L}\sum_{m=1}^{M} a_{u,v}\times T^m \times Y^m_{u,v} + \sum_{s\in S}\sum_{m=1}^{M} b_s \times T^m \times Y^m_s.$$

The objective function is subject to the following constraints:

Flow Conservation Constraint. For flow m, the traffic received on node u (a node can be a switch or a host) is equal to the traffic it sends out, unless it is the source node which generates the flow m, or the destination node which consumes the flow m. The constraint can be given by

$$\sum_{\{v|(u,v)\in L\}} f^m_{u,v} - \sum_{\{v|(v,u)\in L\}} f^m_{v,u} = \begin{cases} \kappa^m, & \text{if } u = src_m, \\ -\kappa^m, & \text{if } u = dst_m, \\ 0, & \text{otherwise} \quad . \end{cases} \tag{15}$$

where $f^m_{u,v}$ represents the size of flow m on link (u,v). $\sum_{\{v|(u,v)\in L\}} f^m_{u,v}$ and $\sum_{\{v|(v,u)\in L\}} f^m_{v,u}$ respectively represent the traffic received and sent by flow m on node u. κ^m, src_m, dst_m represent the bandwidth, source, and destination of flow m, respectively.

Bandwidth Capacity Constraint. Traffic on each link in the network should not exceed link capacity. The constraint can be expressed as

$$\sum_{m=1}^{M} f^m_{u,v} \le c_{u,v} \times \sum_{m=1}^{M} Y^m_{u,v}, \tag{16}$$

where $c_{u,v}$ indicates the capacity of link (u,v).

Relationship Constraints. We assume that the flow is indivisible, which implies that the size of flow m on any link (u,v) is either κ^m or 0. Then, a binary variable, $X^m_{u,v}$, indicating whether flow m passes through link (u,v) can be expressed as

$$X^m_{u,v} = \frac{f^m_{u,v}}{\kappa^m}. \tag{17}$$

If a flow passes through a link connected to switch s, then the flow passes through switch s. Otherwise if a flow does not pass through any link connected to switch s, then the flow does not pass through switch s. Thus the constraint can be expressed by

$$\begin{aligned} X^m_s &\ge X^m_{s,v}, \{\forall v|(s,v)\in L\} \\ X^m_s &\le \sum_{\{v|s,v\in L\}} X^m_{s,v}, \end{aligned} \tag{18}$$

where X^m_s is a binary variable indicating whether flow m passes through switch s.

$Y^m_{u,v}$ indicates whether flow m is the flow with longest duration on link (u,v). If flow m is the flow with longest duration, $Y^m_{u,v}$ is 1, otherwise 0, which can be expressed by

$$Y^m_{u,v} = \begin{cases} 1, & \text{if } m = \arg\max_{\forall j}\{T_j \times X^j_{u,v}\}, \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

Similarly, Y^m_s indicates whether flow m is the flow with longest duration on switch s, which can be expressed as

$$Y^m_s = \begin{cases} 1, & \text{if } m = \arg\max_{\forall j,\forall v|(s,v)\in L}\{T_j \times X^j_{s,v}\} \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

Once the routing path of flow m contains sleeping switches, these switches must be activated before transmit flow m. Only if all switches on the routing path of flow m are activated, no switches need to be activated. The relationship constraint can be expressed as

$$\begin{aligned} \alpha^m &\le \sum_{s\in S} X^m_s \times A_s \\ \alpha^m &\ge X^m_s \times A_s, \forall s \in S, \end{aligned} \tag{21}$$

where A_s is a binary constant indicating whether switch s is asleep. If switch s is asleep, then A_s is 1; otherwise, it is 0. α^m is a binary variable indicating whether sleep switches are included on the routing path of flow m. If the routing path of flow m contains sleeping links, then α^m is 1; otherwise 0.

Similarly, once the routing path of flow m contains sleeping links, these links must be activated before transmit flow m. Only if all links on the routing path of flow m are activated, no links need to be activated. The relationship constraint can be expressed as

$$\begin{aligned} \beta^m &\le \sum_{(u,v)\in L} X^m_{u,v} \times A_{u,v} \\ \beta^m &\ge X^m_{u,v} \times A_{u,v}, \forall (u,v) \in L, \end{aligned} \tag{22}$$

where $A_{u,v}$ is a binary constant indicating whether link (u,v) is asleep. If link (u,v) is asleep, then $A_{u,v}$ is 1; otherwise, it is 0. β^m is a binary variable indicating whether sleep links are included on the routing path of flow m. If the routing path of flow m contains sleeping links, then β^m is 1, otherwise 0.

If the forwarding rules of flow m already exist in the network and does not need to be modified, then no new forwarding rules need to be installed; otherwise, new forwarding rules need to be installed. The constraint can be represented as

$$\gamma^m = \begin{cases} 0, & \text{if exist a rule for flow } m \text{ and no modify}, \\ 1, & \text{otherwise.} \end{cases} \tag{23}$$

where γ^m is a binary variable indicating whether flow m needs to update rules. If flow m needs to install new forwarding rules, γ^m is 1; otherwise 0.

4.4 NP-Complete Proof

In this subsection, we prove that the JMDRD problem is NP-complete.

Theorem 1. *The JMDRD problem belongs to NP.*

Proof. Given a solution of the JMDRD problem (a path from the source to the destination), we can verify whether the path satisfies the constraints of the JMDRD problem in polynomial time. Therefore, the JMDRD problem belongs to NP. □

Theorem 2. *The JMDRD problem is NP-hard.*

Proof. Here we prove that the JMDRD problem is NP-hard by reducing the JMDRD problem to Bin-Packing problem.

Bin-Packing problem can be described as: Given a set of bins and items, each bin has a capacity and each item has a size. We need to determine which bin to pack witch item so that the items in each bin do not exceed the capacity while the number of bins used is the least. Let y_i indicate whether bin i is used and $X_{i,m}$ denote whether item m is packed in bin i. Let M and N denote the number of items and bins, respectively. Let w_m represent the size of item m, and c_i denote the capacity of bin i. The problem can be formulated as follows:

$$\min \sum_{i=1}^{N} y_i$$
$$s.t. \begin{cases} \sum_{m=1}^{M} w_m x_{i,m} \le c_i y_i, \\ \sum_{i=1}^{N} x_{i,m} = 1, \\ x_{i,m} \in \{0,1\}, \quad i = 1,\ldots,N, m = 1,\ldots,M. \\ y_i \in \{0,1\}, \quad i = 1,\ldots,N. \end{cases} \tag{24}$$

The Bin-Packing problem is a classic NP-complete problem. In the following, we reduce the JMDRD problem to the Bin-Packing problem.

Considering a topology as shown in Fig. 3. There are M flows in the network, in which each flow is from $h_m^{'}$ to h_m, has a bandwidth κ^m and data size r^m. All flows have the same data transmission duration T_{data}, i.e., $\frac{r^m}{\kappa^m} = T_{data}$. The capacity of link $(h_m^{'}, u)$ and (v, h_m) is set as ∞. The power of each link and each switch is a and b, respectively. All network devices are activated. Installing forwarding rules for a flow takes time τ. In addition, there are no forwarding rules are installed on the switches in the network. Let

$$T_{total} = T_{data} + \tau$$
$$c_i = \min\{c_{u,i}, c_{i,v}\}$$
$$z_i = \sum_{m=1}^{M} Y_i^m = \sum_{m=1}^{M} Y_{s,i}^m = \sum_{m=1}^{M} Y_{i,d}^m.$$

Then, the JMDRD problem becomes

$$\min \sum_{i=1}^{N} (2a+b) \times T_{total} \times z_i + (2Ma + 2b) \times T_{total}$$
$$s.t. \begin{cases} \sum_{m=1}^{M} \kappa^m x_{i,m} \le c_i z_i, \\ \sum_{i=1}^{N} x_{i,m} = 1, \\ x_{i,m} \in \{0,1\}, \quad i = 1,\ldots,N, m = 1,\ldots,M. \\ z_i \in \{0,1\}, \quad i = 1,\ldots,N. \end{cases} \tag{25}$$

Obviously, problem (25) is equivalent to problem (24), thus the JMDRD problem is reduced to the Bin-Packing problem. This completes the proof. □

According to the above two theorems, the JMDRD problem is NP-hard, and belongs to NP. Therefore, the JMDRD problem is NP-complete. In general, solving NP-complete problems is time consuming, especially in large-scale data centers where a large number of flows need to be processed. However, latency is one of the most important performance metrics in the data center. Thus we will propose a heuristic algorithm with low time complexity to solve the JMDRD problem.

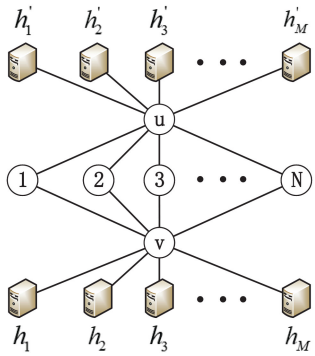


Fig. 3. A special example of the JMDRD problem.

5 Algorithm Description

In this section, we propose a heuristic Green Network (GN) algorithm, which aims to provide the minimum energy consumption path for each flow by finding the least weighted path in a weighted graph. We first introduce how to construct the weighted graph for each flow.

5.1 Build Weighted Graph

The weighted graph $G' = (V', L', w)$ is constructed from $G = (V, L)$ for flow m as follows. Link $(u, v) \in L$ is added to L' if the remaining bandwidth of the link can meet the bandwidth requirement of flow m. Link $(u, v) \in L'$ is assigned a weight $w_{u,v}$, which is positively related to the energy consumed by a flow on a link. In other words, the more energy a flow needs to consume through a link, the greater the weight of the link on the weighted graph. The value of $w_{u,v}$ can be given by

$$w_{u,v} = a_{u,v} \times T_{u,v}^m + \tfrac{1-\phi_u^m}{2} b_u \times T_u^m + \tfrac{1-\phi_v^m}{2} b_v \times T_v^m, \tag{26}$$

where $w_{u,v}$ denotes the weight of link (u, v), and ϕ_u^m denotes whether node u is the source or destination of flow m. If node u is the source or destination of flow m, then ϕ_u^m is 1; otherwise 0.

Lemma 1. *In a weighted graph for an arbitrary flow, the energy consumed by the flow passing a path is equal to the weight of the path.*

Proof. Suppose there is a flow m in the network G, G' is the weighted graph constructed for flow m, and path p is an arbitrary path planned for flow m in G'. We define $\mathbb{k}$ represent the switches on path p of flow m. The energy consumption of flow m on path p is

$$E^m = \sum_{u,v \in p} T_{u,v}^m \times a_{u,v} + \sum_{s \in \mathbb{k}} T_s^m \times b_s. \tag{27}$$

It should be noted that we only calculate network energy consumption and do not take into account server energy consumption, such as energy consumption of source and destination nodes.

The weight of path p is equal to the sum of the weights of all links on path p. According to Equation (26), the weight of path p can be expressed as

$$W_p^m = \sum_{u,v \in p} w_{u,v} = \sum_{u,v \in p} T_{u,v}^m \times a_{u,v} + \sum_{u \in \mathbb{k}} T_u^m \times b_u. \tag{28}$$

This completes the proof. □

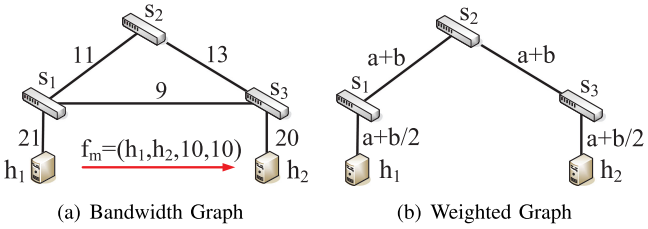


Fig. 4. Example of establishing weighted Graph.

In the following, we present an example to illustrate the construction of weighted graph. As shown in Fig. 4a, the remaining bandwidth for a link is marked on the link. There is a traffic demand $f_m = \{h_1, h_2, 10, 10\}$ arrives in the network, where h_1, h_2, 10 and 10 means the source, destination, bandwidth, and data size of flow f_m, respectively. We assume that all devices have been activated and the forwarding rules for flow f_m have been installed. According to Equation (7), we can get the duration of flow f_m to be 1 s. In addition, we assume that the duration of all devices is zero before flow f_m is added, and the power of each link and switch is a and b, respectively. According to the above construction method of the weighted graph, we can get the weighted graph for flow f_m as shown in Fig. 4b, where the weight of each link is marked. We can plan a path for flow f_m, for example, $h_1 - s_1 - s_2 - s_3 - h_2$, which has a weight of $(4a + 3b) \times t$. We can find that once the flow f_m is transmitted on this path, the duration of 3 switches and 4 links need to be extended by 1 s, which means that the transmission of flow f_m needs to consume network energy of $(4a + 3b) \times t$. Obviously, the weight of the path planned for the flow f_m is equal to the energy that the flow f_m consumes on that path. That is, the greater the weight of a path, the more energy the flow consumes through the path. Note that the weighted graph construction method described in this subsection will be represented by the $WeightedGraph()$ function.

5.2 Algorithm Design

In this subsection, we describe the Green Network (GN) algorithm in detail. Specifically, we first introduce Algorithms 1 and 2, which are used to update network resources and plan paths for flows, respectively. Subsequently, we introduce Algorithm 3 to achieve the flow scheduling.

5.2.1 Resource Updating

The resource update algorithm is used to update information about the duration of devices and the remaining bandwidth of links. It provides real-time information for the path planning of Algorithm 2, which ensures that the traffic on each link does not exceed the capacity and Algorithm 2 has enough information to construct a weighted graph. In the real network, there are two situations that need to update the network information, that is, the resource needs to be occupied by a newly arrived flow or the resource needs to be released by a previous flow.

Specifically, as shown in Algorithm 1, if a new flow f_m arrives and needs to occupy network resources, then the remaining bandwidth of each link on its routing path is reduced. In addition, the duration of the links and switches on the routing path of flow f_m is updated. Once the duration of a link (switch) on the routing path is shorter than the duration of the flow f_m, the duration of the link (switch) is prolonged by flow f_m, otherwise the duration of the link (switch) remains unchanged. If a previous flow f_m completes the transmission and needs to release the occupied network resources, then the remaining bandwidth of each link on its routing path is increased. In addition, the duration of the links and switches on the routing path of flow f_m is also updated. Once there is no active flow passes a link (switch), the duration of the link (switch) is zero. Otherwise, the duration of the link (switch) is equal to the duration of the active flow that passes through the link (switch) and has the longest duration.

Algorithm 1. Resource Updating

Input: F_{active}: set of active flows; $\mathcal{C}$: link remaining capacity; T: duration of link, switch and flow; f_m: traffic demand; $state$: indicate occupy or release resources.
Output: $T, \mathcal{C}$

if $state == 'occupy'$ **then**
  for each link $(u, v) \in f_m.path$ **do**
    $\mathcal{C}_{u,v} = \mathcal{C}_{u,v} - \kappa^m$;
    $T_{u,v} = \max\{T_{u,v}, T^m\}$;
  end for
  for each switch $s \in f_m.path$ **do**
    $T_s = \max\{T_s, T^m\}$;
  end for
end if
if $state == 'release'$ **then**
  for each link $(u, v) \in f_m.path$ **do**
    $\mathcal{C}_{u,v} = \mathcal{C}_{u,v} + \kappa^m$;
    if $\nexists f_i \in F_{active}$ passes link (u, v) **then**
      $T_{u,v} = 0$;
    else
      $T_{u,v} = \max_{\forall f_i \in F_{active}, (u,v) \in f_i.path}\{T^i\}$;
    end if
  end for
  for each switch $s \in f.path$ **do**
    if $\nexists f_i \in F_{active}$ passes switch s **then**
      $T_s = 0$;
    else
      $T_s = \max_{f_i \in F_{active}, s \in f_i.path}\{T^i\}$;
    end if
  end for
end if

5.2.2 Path Planning

As analyzed in the Section 3, the energy consumption for switch activation is not negligible for both short flow and long flow. The energy consumption for rule installation and link activation account for a significant portion of the total energy consumption of the short flow, however, it is negligible for the long flow. Thus, in the Algorithm 2, short flow and long flow are considered separately.

In the Algorithm 2, on the one hand, for a short flow, the algorithm first attempts to forward the flow using the existing forwarding rules in the network. In this case, the flow does not need to activate devices and install rules so that the flow can complete the transfer faster. If there is no forwarding rule for the flow in the network or there is not enough bandwidth on the forwarding path, then the algorithm attempts to find a path in the network that contains only the activated devices. Specifically, the path planning algorithm first removes the

sleep links and sleep switches from the graph G to obtain a subgraph G_s, and then constructs a weighted graph for G_s using the method in the Section 5.1, and finally finds a minimum weighted path from the weighted graph G' by Dijkstra algorithm to find a minimum energy consumption path. In this case, the forwarding rules need to be installed before the flow is transmitted.

Algorithm 2. Path Planning

Input: G: network topology; $\mathcal{C}$: link remaining capacity; T: duration of link, switch and flow; f_m: traffic demand.
Output: $f_m.path$

if f_m is $shortflow$ **then**
  if f_m exist a path in $paths$ and requested bandwidth is satisfied **then**
    $f_m.path = paths[f_m]$
  else
    $G_s \leftarrow$ remove sleep links and sleep switches from G
    $G' = WeightedGraph(G_s, T, \mathcal{C}, f_m.src, f_m.dst)$
    $f_m.path = G'.dijkstra(f_m.src, f_m.dst)$
    if $f_m.path == NOPATH$ **then**
      $G_s \leftarrow$ remove sleep switches from G
      $G' = WeightedGraph(G_s, T, \mathcal{C}, f_m.src, f_m.dst)$
      $f_m.path = G'.dijkstra(f_m.src, f_m.dst)$
      if $f_m.path == NOPATH$ **then**
        $G' = WeightedGraph(G, T, \mathcal{C}, f_m.src, f_m.dst)$
        $f_m.path = G'.dijkstra(f_m.src, f_m.dst)$
      end if
    end if
  end if
else
  $G_s \leftarrow$ remove sleep switches from G
  $G' = WeightedGraph(G_s, T, \mathcal{C}, f_m.src, f_m.dst)$
  $path = G'.dijkstra(f_m.src, f_m.dst)$
  if $path == NOPATH$ **then**
    $G' = WeightedGraph(G, T, \mathcal{C}, f_m.src, f_m.dst)$
    $path = G'.dijkstra(f_m.src, f_m.dst)$
  end if
end if

If the routing path of the flow cannot be found in the network that has no sleep switches and links, then the algorithm attempts to plan a path for the flow in the network that has no sleep switches. In this case, the sleep links on the routing path need to be activated and the forwarding rules need to be installed before the transmission of the flow. If the path cannot be found yet, the algorithm attempts to find a path for the flow in the network that containing sleep switches and sleep links.

On the other hand, for long flow, the algorithm tries to find a minimum energy consumption path in the network that has no sleep switches. If this path cannot be found, then the algorithm tries to find a minimum energy consumption path for the flow in the whole network.

5.2.3 Greening Network Algorithm

Algorithm 3 is triggered when a flow finishes transmission or a new flow arrives. On the one hand, if a new flow arrives, Algorithm 2 is called to plan a path for the flow. If no path can be found in the network to meet the demand of the flow, the flow is suspended. Otherwise, the assigned path is added to the set of paths and the flow is added to the set of activated flows. Then, the controller sends commands to activate sleeping devices on the routing path, and the switches install the forwarding rules sent from the controller. Finally, Algorithm 1 is called to update the bandwidth resources of the network, as well as the duration of links and switches.

On the other hand, if a flow completes the transmission, then the flow is removed from the set of activated flows. Algorithm 1 is then called to release the bandwidth resource occupied by the flow on the routing path and update the duration of the devices on the routing path. Since the occupied bandwidth resources are released, the suspended flows can be judged one by one whether the resources in the network can satisfy the transmission request of the flow. The suspended flow can be activated if its request can be satisfied.

Algorithm 3. Green Network Algorithm

Input: G: network topology; F_{active}: set of active flows; $F_{suspend}$: set of suspended flows; f_m: a new arriving flow or a finished flow; $\mathcal{C}_{u,v}$: link remaining capacity; T: duration of link, switch and flow.
Output: path for each flow

if a flow f_m is $arrives$ **then**
  $f_m.path = PlanningPath(G, f_m, T)$;
  if $f_m.path == NOPATH$ **then**
    $F_{suspend} = F_{suspend} + \{f_m\}$;
  else
    $paths[f_m] = f_m.path$;
    $F_{active} = F_{active} + \{f_m\}$;
    Activate devices, install rules;
    $ResourceUpdating(F_{active}, \mathcal{C}_{u,v}, T, f_m, 'occupy')$;
  end if
end if
if a flow f_m is $finishes$ **then**
  $F_{active} = F_{active} - \{f_m\}$;
  $ResourceUpdating(F_{active}, \mathcal{C}_{u,v}, T, f_m, 'release')$;
  Schedule suspended flows in $F_{suspend}$;
end if

5.3 Algorithm Complexity Analysis

In this subsection, we analyze the complexity of the GN algorithm. Assume that there are n nodes, m flows and l links in the network. For Algorithm 1, if a new flow arrives and needs to occupy resources, in the worst case, the algorithm complexity is $O(2n + 3l)$. If a flow completes transmission to release resources, in the worst case, the complexity of Algorithm 1 is $O(2mn^2 + 2ml^2)$. For short and long flow, in the worst case, the complexity of Algorithm 2 is $O(m + 6l + 3n \log n)$ and $O(4l + 2n \log n)$, respectively. According to the above analysis, we can draw a conclusion: in the worst case, for the arrival of short flow, the arrival of long flow, and the completion of flow transmission, the complexity of Algorithm 3 is $O(m + 9l + 3n \log n)$, $O(7l + 2n \log n)$ and $O(2mn^2 + 2ml^2)$, respectively.

6 Performance Evaluation

In this section, we first introduce the simulation settings and then explore the performance gap between the GN algorithm and the optimal solution. Finally, we compare the GN algorithm with the two network energy-saving algorithms

under different network topologies, network sizes, traffic loads and parameter settings.

6.1 Simulation Settings

Operating Environment. In our experiments, the operating system is Ubuntu 14.04 and the programming language is Python 2.7. The hardware configuration is: 4G RAM, Intel Core i7-5557U processor with 3.10 GHz.

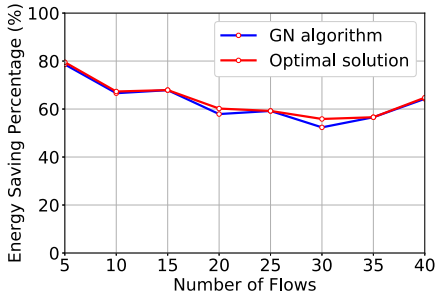
Topology. The GN algorithm is implemented in four topologies. The first one is the small-scale Fat-Tree topology [29], where each switch has 4 ports. In this topology, the number of core switches, aggregation switches, edge switches, and hosts is 4, 8, 8, and 16, respectively. The second one is the Bcube topology [30], which is composed of 16 hosts and 2 levels of switches. The third one is the VL2 topology [31] with 2 core switches, 8 aggregation switches, and 8 ToR switches and 16 hosts. The last one is large-scale Fat-Tree topology with 16 pods, in which the number of core switches, aggregation switches, edge switches and hosts is 64, 128, 128 and 1024, respectively.

Parameter Setting. The experimental parameters are given as follows. The power of each switch and each link is 48 and 4 W [32], respectively. The capacity of each link is set to 1 Gbps [13]. The boundary between the long flow and the short flow is set to 1 Mbps [5], [31], [33]. The time required to activate the link and switch is set to 1,000 and 10 ms [7], [13], [22]. The duration of the rule installation is set to 10 ms [24]. In order to simplify the experiment, we assume that there are no forwarding rules in the network and all network devices are sleeping.

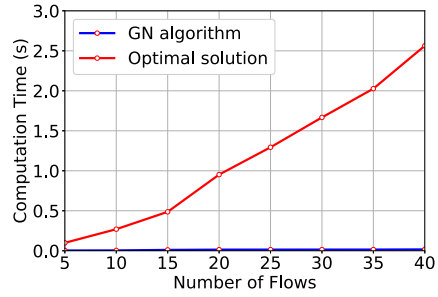
Work Load. In our trace-driven experiment, the traffic load is generated based on a real Internet traffic matrix [15], [34]. The traffic matrix is collected from a 23-nodes network for 4 months and contains information about each flow, such as its source, destination, and bandwidth. The duration of a flow obeys the Poisson distribution [35]. The data size of a flow is equal to the transmission duration of the flow multiplied by the required bandwidth. Different numbers of flows are used to simulate various network loads. However, there are far more hosts in the large-scale Fat-Tree network than hosts in that traffic matrix. We generate the source, destination, and bandwidth of each flow in the large-scale Fat-Tree network as follows: The event that the source and destination of a flow lie in the same rack obey the Bernoulli distribution. The bandwidth required for a flow is subject to the Gaussian distribution [35].

Comparison Algorithm. We compare GN algorithm with the following algorithms:

- FCTcon algorithm: The FCTcon algorithm reduces network power through traffic aggregation and optimizes flow completion time through dynamic flow bandwidth control. Power and flow completion time are jointly considered to save network energy consumption.
- DM algorithm: The Dijkstra-Modify (DM) algorithm uses the Dijkstra algorithm as the path planning strategy and sets unnecessary network devices to sleep to save energy.
- Optimal solution: The optimal solution is obtained by solving our JMDRD problem with the optimization tool Gurobi [36].



(a) Performance Comparison on Energy Consumption



(b) Performance Comparison on Computation Time

Fig. 5. Performance comparison between GN algorithm and optimal solution.

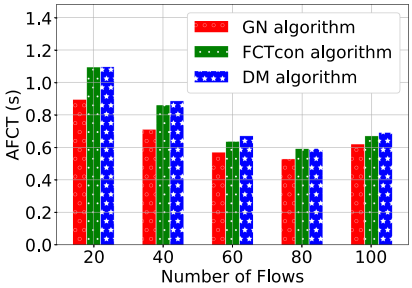
6.2 Comparison between GN Algorithm and Optimal Solution

In this subsection, we evaluate the gap between our GN algorithm and optimal solution in a small-scale Fat-Tree topology. In the network, different numbers of flows are used to simulate different network loads. Fig. 5 depicts the performance comparison of the GN algorithm and the optimal solution in terms of energy saving percentage and computation time. The energy saving percentage mentioned here refers to the energy saving percentage of the GN algorithm compared to the Dijkstra algorithm.

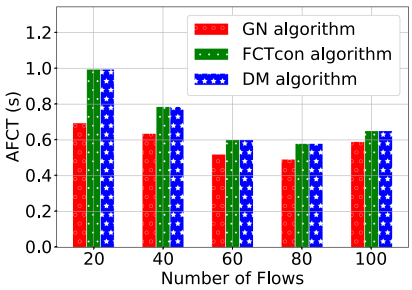
As shown in Fig. 5a, the energy saving performance of GN algorithm can well approximate the optimal solution. In the worst case, i.e., there are 30 flows, the energy saving percentage gap between the GN algorithm and the optimal solution is only 3.5 percent. In other words, the largest gap between the GN algorithm and the optimal solution is not more than 3.5 percent. In addition, the energy saving percentage of the GN algorithm is no less than 52 percent. As shown in Fig. 5b, the GN algorithm consumes significantly less time to plan paths for all flows than to solve the optimal solution. When the number of flows to be planned is 40, the GN algorithm only needs 50 ms to plan the paths for all flows (each flow consumes less than 1.3 ms), while it takes 2.56 s to solve the optimal solution. Therefore, we can conclude that the energy saving performance of the GN algorithm is close to the optimal solution, and it runs faster.

6.3 Performance in Different Network Topology

In this part, we test the Average Flow Completion Time (AFCT) and FCT cumulative distribution performance of the GN algorithm in the Fat-Tree topology and the Bcube topology, as shown in Figs. 6 and 7. More importantly, we



(a) AFCT performance in Fat-Tree



(b) AFCT performance in Bcube

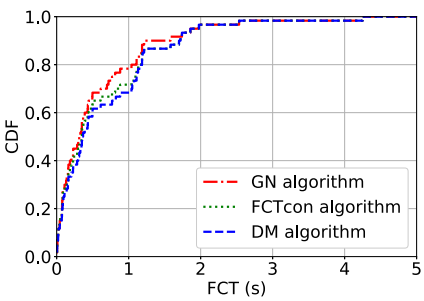
Fig. 6. The AFCT performance comparison of GN algorithm, FCTcon algorithm and DM algorithm in Fat-Tree and Bcube networks.

evaluate the energy performance of the GN algorithm in Fat-Tree topology, Bcube topology and VL2 topology, as shown in Fig. 8.

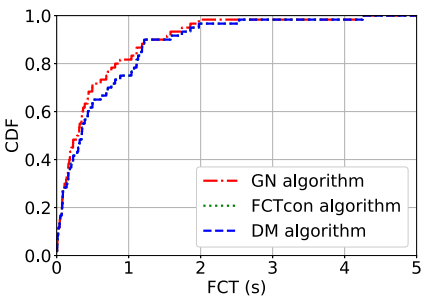
6.3.1 Average Flow Completion Time

As shown in Fig. 6a, in Fat-Tree networks, the AFCT of the GN algorithm is always lower than that of the FCTcon algorithm and the DM algorithm. For example, when the number of flows is 40, the GN algorithm can reduce AFCT by 17.43 and 19.80 percent compared to the FCTcon algorithm and the DM algorithm, respectively. Besides, compared with the FCTcon algorithm and the DM algorithm, the GN algorithm can reduce the AFCT by 13.75 and 15.55 percent on average. This is because device activation and rule installation operations are optimized in the GN algorithm, which reduces the time taken to prepare the flow for transmission. The FCTcon algorithm reduces energy consumption by traffic aggregation, that is, aggregating flows to activated network devices, which reduces the operation of device activation as well as the AFCT of the flow. Therefore, the AFCT performance of the FCTcon algorithm is shorter than that of the DM algorithm and is closer to that of our GN algorithm.

As shown in Fig. 6b, compared with the FCTcon algorithm and the DM algorithm, the GN algorithm reduces the average AFCT by 18.87 percent. Besides, the FCTcon algorithm and the DM algorithm have the same AFCT performance. This is because, in the Bcube network, the number of hops between servers is significantly reduced than that in the Fat-Tree

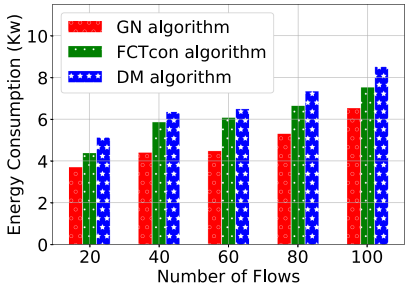


(a) Fat-Tree

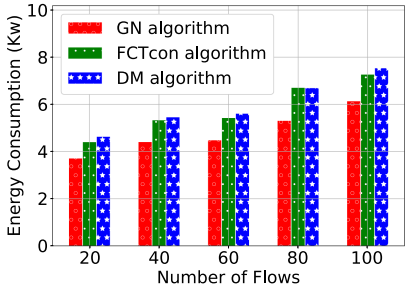


(b) Bcube

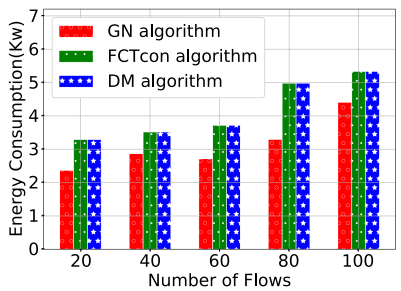
Fig. 7. The FCT distribution comparison of GN algorithm, FCTcon algorithm and DM algorithm in Fat-Tree and Bcube networks.



(a) Fat-Tree



(b) VL2



(c) Bcube

Fig. 8. The energy consumption performance of GN algorithm, FCTcon algorithm and DM algorithm in Fat-Tree and Bcube networks.

network, the fewer paths can be used as candidate paths, and the energy consumption gap between these candidate paths is significant. As the number of flows increases, the AFCT performance gap between GN algorithm, FCTcon algorithm and DM algorithm becomes smaller. This is because the increased flows share the cost of device activation.

6.3.2 Distribution of FCT

As shown in Fig. 7a, in the Fat-Tree network, the flows scheduled by GN algorithm can always complete the transmission before the flow scheduled by FCTcon algorithm and DM algorithm. For example, 78.33 percent of the flows scheduled by GN algorithm can complete transmission within 1 s, while only 71.67 and 68.33 percent of the flows scheduled by FCTcon and DM algorithm can complete the transmission. As shown in Fig. 7b, in Bcube topology, compared with the FCTcon algorithm and the DM algorithm, the flows scheduled by the GN algorithm also always complete transmission earlier. In addition, the FCTcon algorithm and DM algorithm have the same FCT performance. In the GN algorithm, 81.67 percent of the flows can be transmitted within 1 s, while in the FCTcon algorithm and the DM algorithm, only 75 percent of the flows complete the transmission within 1 s.

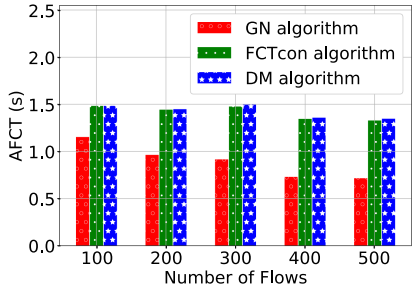
6.3.3 Energy Consumption

We can observe from Fig. 8a that the GN algorithm always consumes less energy than the FCTcon algorithm and the DM algorithm. Specifically, the GN algorithm can reduce energy consumption by 13.61 and 26.87 percent on average compared to the FCTcon algorithm and the DM algorithm. This is because the GN algorithm optimizes the energy required for each flow to prepare for transmission, depending on the characteristics of long and short flow, thus it consume less rule installation energy consumption and device activation energy consumption. In addition, the energy consumed by the three algorithms increases as the number of flows increases, as more bits need to be transmitted. Similar energy consumption results can be found in Fig. 8b, since the network structure between VL2 and Fat-Tree is similar. In addition, the three algorithms consume slightly less energy in the VL2 network than in the Fat-Tree network. Because there are more redundant links and fewer switches in VL2, the power of the switch is significantly higher than the power of the link.

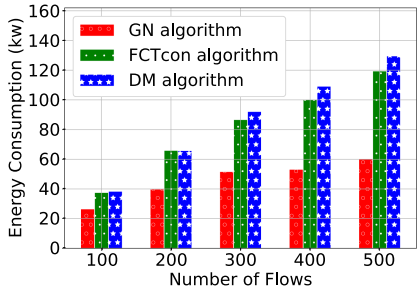
As shown in Fig. 8c, in the Bcube network, the FCTcon algorithm and the DM algorithm have the same energy consumption performance. This is because there are few alternate redundant paths for FCTcon algorithm to aggregate traffic, in Bcube networks. More importantly, the GN algorithm always consumes less energy than the FCTcon algorithm and the DM algorithm. When the number of flows is 100, GN algorithm can reduce energy consumption by 24.89 percent, compared with FCTcon algorithm and DM algorithm.

6.4 AFCT and Energy Consumption in Large Scale Network

In this subsection, we evaluate the performance of the GN algorithm in a large-scale Fat-Tree network. The number of flows varies from 100 to 500 to simulate the traffics in the network. Fig. 9 depicts the performance comparison between the GN algorithm, the FCTcon algorithm and the DM algorithm in terms of the Average Flow Completion Time (AFCT) and energy consumption.



(a) AFCT Performance Comparison



(b) Performance Comparison of Energy Consumption

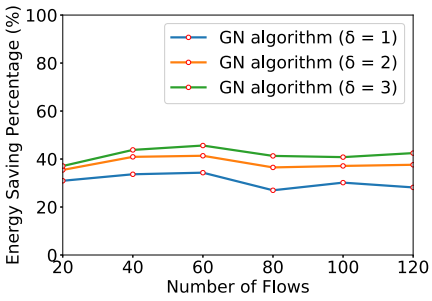
Fig. 9. The performance of GN algorithm, FCTcon algorithm and DM algorithm in the large-scale Fat-Tree network.

We can observe from Fig. 9a that, in large scale network, the AFCT of GN algorithm is still shorter than that of FCTcon algorithm and DM algorithm. Compared with the FCTcon algorithm and the DM algorithm, the GN algorithm can reduce AFCT by 36.61 and 37.11 percent on average. Obviously, the GN algorithm has a greater advantage in large-scale networks. Because there are more redundant links and switches in large-scale networks, this provides more opportunities for optimizing device activation and rule installation.

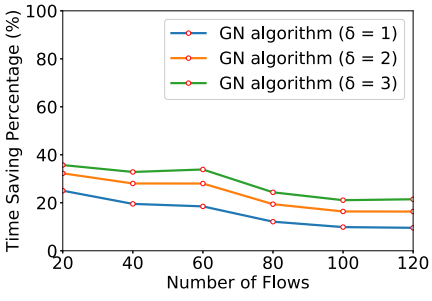
We can also find from Fig. 9b, the GN algorithm is superior to the FCTcon algorithm and the DM algorithm in terms of energy consumption in large-scale networks. Compared with FCTcon algorithm and DM algorithm, our GN algorithm can reduce energy consumption by 43.47 and 46.78 percent on average. This result shows that the GN algorithm has a more obvious energy-saving advantage in large-scale networks, as more devices and more flows provide more opportunities for energy conservation. In addition, the energy consumption of the three algorithms in large-scale networks is higher than that of small-scale networks, because more flows need to be transmitted, and more devices need to be run.

6.5 The Performance of GN Algorithms Under Different Switch Activation Duration

In this subsection, we test the energy saving percentage and time saving percentage performance of the GN algorithm with different switch activation duration. It is worth noting that the



(a) Energy Performance



(b) Time Performance

Fig. 10. The performance of GN algorithms under different switch activation duration.

energy saving percentage and time saving percentage mentioned here is the GN algorithm compared to the DM algorithm. In the experiment, the switch activation duration δ is set to 1, 2 and 3 s. The simulation results are shown in Fig. 10.

We can find from Fig. 10a that the GN algorithm can always save energy compared to the DM algorithm. Even in the worst case, i.e., the number of flows is 20 and $\delta = 1$, the GN algorithm can save 26.97 percent energy. In addition, the energy saving of the GN algorithm with $\delta = 3$ can always be better than that of the GN algorithm wih $\delta = 1$. This is because the greater the activation duration of switch, the greater the ratio of device activation energy consumption to total energy consumption. Then, optimizing the device activation energy consumption can bring greater advantages.

We can also observe from Fig. 10b that the GN algorithm saves FCT compared to the DM algorithm. In the worst case, i.e., $\delta = 1$ and the number of flows is 100, the GN algorithm can shorten the FCT by 9.57 percent. In addition, the GN algorithm with $\delta = 3$ is superior to the GN algorithm with $\delta = 1$ in terms of time saving percentage. This is because the greater the value of δ is, that is, the longer the switch activation duration is, the larger the switch activation duration taking up in total FCT, and the more FCT can be saved by optimizing the operation of device activation.

6.6 The Computation Time Performance of GN Algorithms Under Different Network Size

In this subsection, we evaluate whether the computation time of GN algorithm is negligible for each flow. We define $T^m_{computation}$ to indicate the computation time of GN algorithm planning a path for an arbitrary flow m. Then our primary metric, which is referred to as computation time percentage, denoted by CTP , can be computed as, $CTP = \frac{T^m_{excution}}{T^m}$. We evaluate the GN algorithm in different size of Fat-Tree networks, where the number of pods is set to 4, 6, 8, 10, and the number of hosts is 16, 54, 128, 250, 432. Each value in the graph is obtained by averaging 100 flows. The default path for each flow in the network is installed.

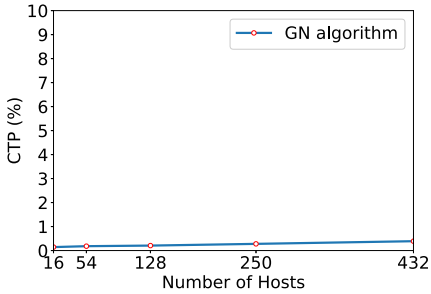


Fig. 11. The CTP performance of GN algorithm under different network sizes.

As shown in Fig. 11, The CTP of GN algorithm increases with the expansion of network scale. Because with the expansion of network scale, planning path for a flow becomes more and more complex. Although the CTP of GN algorithm increases with the expansion of network size, the value does not exceed 0.4 percent. Further, in the data center, the controller has stronger computing power, so the GN algorithm runs faster and the CTP is smaller. Therefore, for each flow, the GN algorithm is acceptable and its time consumption is negligible.

7 Conclusion

In this paper, we study the problem of minimum network energy consumption, including three kinds of energy consumption, i.e., device activation energy consumption, rule installation energy consumption and data transmission energy consumption. Specifically, we analyze the advantages of jointly optimizing three kinds of energy consumption. Then, we construct a JMDRD model. Next, we prove that the JMDRD problem is NP-complete. Subsequently, we propose a heuristic GN algorithm, which processes long flow and short flow separately according to their different characteristics. The GN algorithm focuses on finding a path with minimum data transmission energy for long flow and a path with minimum device activation energy and rule installation energy for short flow. Finally, we find that the energy-saving performance of the GN algorithm is close to the optimal solution (no more than 3.5 percent) and has a short computational time. In addition, the GN algorithm proposed in this paper outperforms the existing algorithms in network energy saving for different network topologies, network sizes, and traffic loads.

Acknowledgments

This work was supported by the National Key R&D Program of China (2018YFB0803400), the Fundamental Research Funds for the Central Universities (2019CDYGZD004), the National Natural Science Foundation of China (No. 61772432, 61772433), the Technological Innovation and Application Demonstration Projects of Chongqing (cstc2018jszx-cyztzxX0014).

REFERENCES

[1] P. Delforge, "America's data centers consuming and wasting growing amounts of energy," 2015. [Online]. Available: http://www.nrdc.org/energy/data-center-efficiency-assessment.asp

[2] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," *ACM SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 338–347, 2010.

[3] Y. Zeng, S. Guo, and G. Liu, "Comprehensive link sharing avoidance and switch aggregation for software-defined data center networks," *Future Gener. Comput. Syst.*, vol. 91, pp. 25–36, 2019.

[4] L. Tsai and W. Liao, "StarCube: An on-demand and cost-effective framework for cloud data center networks with performance guarantee," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 235–249, Jan. 2018.

[5] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 267–280.

[6] M. Gupta and S. Singh, "Dynamic ethernet link shutdown for energy conservation on ethernet links," in *Proc. IEEE Int. Conf. Commun.*, 2007, pp. 6156–6161.

[7] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via sleeping and rate-adaptation," in *Proc. USENIX Symp. Netw. Syst. Design Implementation*, 2008, pp. 323–336.

[8] M. Gupta and S. Singh, "Using low-power modes for energy conservation in Ethernet LANs," in *Proc. IEEE Int. Conf. Comput. Commun.*, 2007, pp. 2451–2455.

[9] B. Heller *et al.*, "ElasticTree: Saving energy in data center networks," in *Proc. USENIX Conf. Netw. Syst. Design Implementation*, 2010, pp. 249–264.

[10] H. Wang, Y. Li, D. Jin, P. Hui, and J. Wu, "Saving energy in partially deployed software defined networks," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1578–1592, May 2016.

[11] K. Zheng, Y. Bai, and X. Wang, "FCTcon: Dynamic control of flow completion time in data center networks for power efficiency," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2019.2912969.

[12] A. Alnoman and A. S. Anpalagan, "Computing-aware base station sleeping mechanism in H-CRAN-Cloud-Edge networks," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2019.2893228.

[13] D. Li, Y. Shang, W. He, and C. Chen, "EXR: Greening data center network with software defined exclusive routing," *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2534–2544, Sep. 2015.

[14] G. Xu, B. Dai, B. Huang, J. Yang, and S. Wen, "Bandwidth-aware energy efficient flow scheduling with SDN in data center networks," *Future Gener. Comput. Syst.*, vol. 68, pp. 163–174, 2017.

[15] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 83–86, 2006.

[16] M. Gupta, S. Grover, and S. Singh, "A feasibility study for power management in LAN switches," in *Proc. 12th IEEE Int. Conf. Netw. Protocols*, 2004, pp. 361–371.

[17] R. Wang, Z. Jiang, S. Gao, W. Yang, Y. Xia, and M. Zhu, "Energy-aware routing algorithms in software-defined networks," in *Proc. Int. Symp. A World Wireless Mobile Multimedia Netw.*, 2014, pp. 1–6.

[18] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for network update," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 323–334, 2012.

[19] J. McClurg, H. Hojjat, P. Černỳ, and N. Foster, "Efficient synthesis of network updates," *ACM SIGPLAN Notices*, vol. 50, no. 6, pp. 196–207, 2015.

[20] S. Vissicchio and L. Cittadini, "FLIP the (Flow) Table: Fast lightweight policy-preserving SDN updates," in *Proc. IEEE 35th Annu. IEEE Int. Conf. Comput. Commun.*, 2016, pp. 1–9.

[21] B. Pfaff *et al.*, "Openflow switch specification, version 1.3.0," Open Networking Foundation, Tech. Rep., 2012.

[22] "IEEE P802.3az energy efficient ethernet task force," 2019. [Online]. Available: https://www.ieee802.org/3/az/index.html

[23] X. Jin *et al.*, "Dynamic scheduling of network updates," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 539–550, 2014.

[24] H. Xu *et al.*, "Joint route selection and update scheduling for low-latency update in SDNs," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3073–3087, Oct. 2017.

[25] X. Zheng, A. Mudambi, and M. Veeraraghavan, "FRTP: Fixed rate transport protocol – a modified version of SABUL for end-to-end circuits," in *Proc. 1st Workshop Provisioning Transport Hybrid Netw.*, 2004, pp. 1–11.

[26] "Tsunami," 2019. [Online]. Available: http://newsinfo.iu.edu/news/page/normal/588.html

[27] B. Eckart, X. He, and Q. Wu, "Performance adaptive UDP for high-speed bulk data transfer over dedicated links," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, 2008, pp. 1–10.

[28] S. Achleitner, N. Bartolini, T. He, T. La Porta, and D. Z. Tootaghaj, "Fast network configuration in software defined networking," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 4, pp. 1249–1263, Dec. 2018.

[29] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

[30] C. Guo *et al.*, "BCube: A high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 63–74, 2009.

[31] A. Greenberg *et al.*, "VL2: A scalable and flexible data center network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 51–62, 2009.

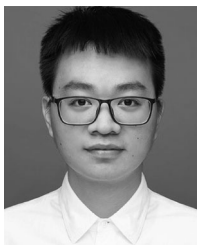
[32] "Cisco Nexus 2200 series data sheet,". 2019. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/switches/ps9441/

[33] J. Zhao, J. Liu, H. Wang, and C. Xu, "Multipath TCP for datacenters: From energy efficiency perspective," in *Proc. IEEE Conf. Comput. Commun.*, 2017, pp. 1–9.

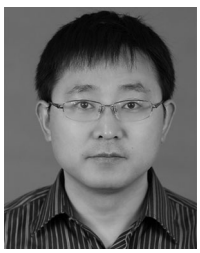
[34] "Traffic Matrices," 2019. [Online]. Available: https://totem.info.ucl.ac.be/dataset.html

[35] S. S. Chandrasekaran, "Understanding traffic characteristics in a server to server data center network," Master's thesis, Rochester Institute of Technology, Rochester, NY, USA, 2017.

[36] "Guribi," 2019. [Online]. Available: http://www.gurobi.com/



Yue Zeng received the BS degree in computer science and engineering from Chongqing Three Gorges University, Chongqing, China, in 2016. He is currently working toward the master's degree in signal and information processing at 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 with Chongqing University and a professor from 2012 to 2018 with Southwest University. He is currently a full professor with Chongqing University, China. He was a senior research associate with the City University of Hong Kong from 2010 to 2011, and a visiting scholar with 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 100 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. He is senior member of the IEEE/ACM.

Guiyan Liu received the BE degree in telecommunications engineering and the MS degree in signal and information processing from Southwest University, Chongqing, China, in 2014 and 2017. She is currently working toward the PhD degree in computational intelligence and information processing at Southwest University. Her research interests include traffic measurement in data center networks and software defined networking.

Pan Li received the BS degree in computer science and engineering from Chongqing Three Gorges University, Chongqing, China, in 2016. She is currently working toward the master's degree in signal and information processing at Southwest University. Her research interests include data center networks and software defined networking.

Yuanyuan Yang received the BEng and MS degrees in computer science and engineering from Tsinghua University, and the MSE and PhD degrees in computer science from Johns Hopkins University. She is a State University of New York (SUNY) distinguished professor of computer engineering and computer science with Stony Brook University, New York, and is currently on leave with the U.S. National Science Foundation as a program manager. Her research interests include wireless networks, data center networks, and cloud computing. She has published more than 400 papers in major journals and refereed conference proceedings and holds seven US patents in these areas. She is currently the associate editor-in-chief of the *IEEE Transactions on Cloud Computing* and an associate editor of the *ACM Computing Surveys*. She has served as an associate editor-in-chief and associated editor of the *IEEE Transactions on Computers* and associate editor of the *IEEE Transactions on Parallel and Distributed Systems*. She has also served as a general chair, program chair, or vice chair for several major conferences and a program committee member for numerous conferences. She is a fellow of the IEEE.