# Online Flow Scheduling with Deadline for Energy Conservation in Data Center Networks

Biyu Zhou*†, Jie Wu‡, Lin Wang§, Fa Zhang¶, and Zhiyong Liu*‖
*Beijing Key Laboratory of Mobile Computing and Pervasive Device, ICT, CAS
†University of Chinese Academy of Sciences, Beijing, China
‡ Center for Networked Computing, Temple University, USA §Technische Universität Darmstadt, Germany
¶Key Lab of Intelligent Information Processing, ICT, CAS
‖State Key Laboratory for Computer Architecture, ICT, CAS
{zhoubiyu, zhangfa, zyliu}@ict.ac.cn, jiewu@temple.edu, wang@tk.tu-darmstadt.de

***Abstract*—We study the problem of flow scheduling in data center networks. Using speed scaling, our aim is to find an online scheduling algorithm that minimizes the total energy consumption of the network by determining both the transmission order and rates of the arriving flows while providing a strict flow deadline guarantee. Observing the superlinear property of link power consumption, the key challenge is in constantly determining the minimum transmission rate for "delay-tolerable" flows without any *priori* knowledge. To leverage the flow arrival pattern, we propose a probability-based flow prediction model to capture the uncertainty of the network flows. Based on the prediction model, we propose a tunable online flow scheduling algorithm to solve the online flow scheduling problem effectively. By introducing a scaling factor on bandwidth allocation, this algorithm allows us to conduct arbitrary trade-offs between the conservative and aggressive behaviors in terms of energy conservation. The effectiveness of the proposed algorithm is validated through rigorous theoretical analysis and further confirmed by extensive numerical simulations.**

***Index Terms*—Data center network, energy efficiency, flow scheduling, online algorithm, traffic prediction.**

## I. Introduction

With the widespread adoption of cloud computing, more and more mega data centers have been deployed in order to match the constantly increasing computing demands. According to [1], the total number of data centers, including various types, will reportedly reach 8.6 million in 2017. As each of the data centers can contain up to hundreds of thousands of servers, the energy consumption of the data centers worldwide has become enormous. Koomey *et al.* predict that the total energy consumption of the data centers worldwide will account for 8% of the total energy consumption for the year in 2020 [2]. Huge energy consumption brings not only a heavy financial burden to data center providers but also serious environmental concerns to the society. The Global e-Sustainability Initiative (GeSI) estimated that by 2020, the greenhouse gas emission from all the data centers will make up 18% of that of the information technology sector [3]. While the servers account for the majority of energy consumption in a typical data center, the network, being one of the major components, also makes a non-negligible impact on the energy footprint, which has been overlooked for a long time [4]. The situation becomes more prominent as sophisticated energy saving techniques are applied to the servers.

In the last decade, there has been a lot of work on reducing the energy consumption in wired networks. Depending on the adopted energy conservation mechanism, these works can be roughly classified into two categories: *power down* [5] and *speed scaling* [6]. The common insight of both mechanisms is to save energy by adjusting the network resource to match the actual workload. Based on the two mechanisms, a number of proposals have been put forward to reduce the energy consumption in data center networks; they focus on network routing [7–10], packet scheduling [11–13], topology design [14, 15], etc. On the other hand, the network's quality of service needs to be ensured when saving energy by shaping the flows. In data center networks, this is usually achieved by guaranteeing flow deadlines [16].

In this paper, a flow is defined as a set of data demands that have to be transmitted from a source to a destination within a given period of time. We consider the problem of optimal flow scheduling for energy efficiency in data center networks using the speed scaling mechanism, while ensuring that flow deadline will not be violated. A similar problem has been studied in [11], but in an offline setting. In contrast, our goal is to solve the online version of the problem which is undoubtedly more challenging. By surveying state-of-the-art solutions, we find that prior solutions fall into at least one of the following flaws: $i$) assuming perfect knowledge about the sequence of flow arrivals, making them impractical [11], $ii$) adopting simple prediction models for future input, resulting in a poor performance in general [17, 18], and $iii$) being devoted to dealing with worst-case inputs, leading to unpredictable average-case performances in practice [19].

We claim that a good solution for flow scheduling in data center networks must possess all the following properties:

1) *Online scheduling*. The input of the flow arrival sequence is revealed over time. The scheduler has to make decisions continuously for the arriving new flows.

This work is supported partially by the National Natural Science Foundation of China (NSFC) Major International Collaboration Project 61520106005, NSFC Project for Innovation Groups 61521092, and the German Research Foundation (DFG) Collaborative Research Center (CRC) 1053 – MAKI.

2) *Incorporating future information*. The future information of flows can be predicted to some extent based on the application characteristics in data centers. The predicted future information should be incorporated in order to improve the quality of the scheduling.
3) *Efficient in practice*. To achieve practical effectiveness, the design of the flow scheduling algorithm should target the average cases rather than devoting themselves to the worst-case inputs, which occur very rarely.

To satisfy all of the above properties, we design a novel online algorithm for flow scheduling in data center networks. The rationale behind our design is to scale up the current resource allocation by a positive factor, thus making room to cope with unpredictable flows in the future. The performance of the algorithm is dominated by this scaling factor, and thus the value for the factor needs to be determined carefully. We will discuss how to tune this value in detail such that the settings will represent the cases with high probabilities (that are likely to occur in practice) and analyze the performance of the online algorithm accordingly. To summarize, we make the following contributions:

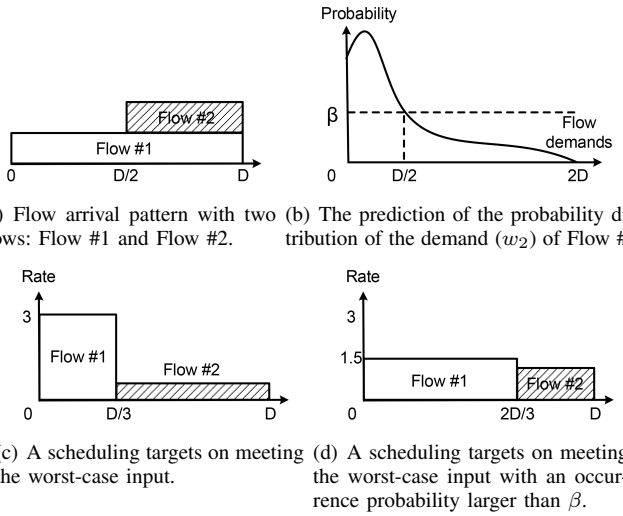
1) We tackle the issue of the deadline-constrained online flow scheduling problem in data centers. We formally define a general probability-based model that captures the uncertainty of the sequentially-revealed flow demands in data center networks.
2) To better exploit the available future knowledge obtained by predictions, we develop a tunable online flow scheduling algorithm for reducing the energy consumption while guaranteeing the flow deadline in data center networks based on the probability-based model.
3) We evaluate our algorithm with comprehensive simulations and experimental results verify that, by leveraging the future information of flow arrivals, the proposed heuristic algorithm can achieve a considerable energy consumption reduction.

The remainder of this paper is organized as follows. Section II provides the motivation for this paper. Section III presents the model for the deadline-constraint, energy-efficient, online flow scheduling (DEFS) problem. The probability-based flow prediction model is also presented in Section III. Section IV explores the general online DEFS problem in a multi-hop setting. Section V presents the numeral evaluation results. Section VII summarizes the related works and Section VII concludes the paper.

## II. Motivation

### A. Flow Scheduling

**Online vs. Offline.** Most previous proposals target an offline setting where entire input sequences are known *a priori* and remain unchanged. The authors in [11] consider an offline setting where the scheduler knows the future, leveraging the characteristic of the application and the history information in data centers. With this advantage, they design an offline scheduling algorithm that achieves the optimal solution. However, actual traffic demands fluctuate dynamically in data centers over time and exhibit both temporal and spatial variations [20]. Thus, accurate values of the entire input sequence are hard to obtain. Even with this prediction, these offline algorithms will perform poorly when the timeline is mismatched.




(a) Flow arrival pattern with two flows: Flow #1 and Flow #2.
(b) The prediction of the probability distribution of the demand ($w_2$) of Flow #2.
(c) A scheduling targets on meeting the worst-case input.
(d) A scheduling targets on meeting the worst-case input with an occurrence probability larger than $\beta$.

Fig. 1. A motivational example.

### B. Prediction and Estimation

**Prediction vs. Non-prediction.** Most previous online scheduling algorithms are either based on very simple models of future flow arrivals or do not assume any future information at all. Without the future flow information, the authors in [19] demonstrate that a myopic online scheduling, which always uses the corresponding optimal decisions at the current time, can have an extremely poor performance under the dynamic traffic scenario. The root cause of this is that myopic online scheduling always serves the fewest demands at a time and leaves the remaining demands for the future because it assumes that no flows will arrive in the future. As time moves towards infinity, the service rate and the energy consumption are unbounded. A scheduling algorithm with a prediction can scale-up the current service rate according to the predictions and will avoid this issue.

**Conservative vs. Aggressive.** Typically, the effectiveness of an online algorithm is evaluated by the competitive ratio, which is the ratio between its worst-case performance and the optimal offline algorithm's performance. While successful in capturing the performance of online algorithms without knowing the exact input, the competitive ratio also suffers from the issue that it measures only "hard" inputs even though "hard" inputs only occur rarely in the real world. This behavior seems conservative. An online algorithm that is designed to obtain a good worst-case competitive ratio also faces the same issue. Considering a scenario where flows arrive over time and the scheduler only knows the exact flow information that is revealed in the current time, due to the superlinear feature of the power function, an optimal schedule will try to balance the transmission rate in each time slot. However, a conservative

scheduler always attempts to serve more flows up-front by setting the current transmission rate to the highest possible value, thus avoiding a potentially large peak in the future. Since these worst-cases are rare in reality, a conservative scheduler performs poorly in real systems.

A motivational example is illustrated in Figure 1. Let $w$ denote the flow demand, which is the amount of data that needs to be routed. Let $r$ and $d$ denote the release time and deadline, respectively. Figure 1(a) depicts a flow arrival pattern with two flows: Flow #1 ($r_1 = 0$, $d_1 = D$, $w_1 = D$) and Flow #2 ($r_2 = D/2$, $d_2 = D$, $w_2$). Figure 1(b) describes a prediction for the probability distribution of the demand $w_2$ of Flow #2. It can be seen that, in the worst case, the flow demand of the flow arriving in the future is predicted to be $2D$ with occurrence probability tending to 0; while this value is estimated to be $D/2$ in the worst case input with an occurrence probability larger than $\beta$. Figure 1(c) illustrates a scheduling targeting for achieving a good performance on the hardest input (almost impossible to happen in real world). Figure 1(d) shows a scheduling that aims to obtain a good performance on the hardest input with an occurrence probability larger than $\beta$ (on behalf of the general situation). It can be seen that, taking into account the fact that the value of the flow demand in a vast majority of cases is less than $D/2$, the transmission rate of the scheduling in Figure 1(d) is much more balanced than that of the scheduling in Figure 1(c). Since the energy consumption function is superlinear in terms of the transmission rate, the scheduling in Figure 1(d) is better than that of Figure 1(c).

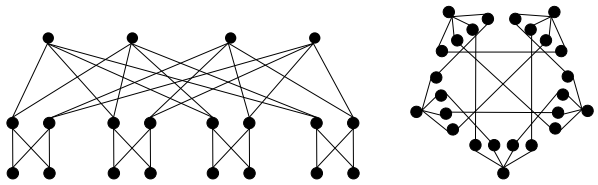
A more aggressive scheduling algorithm that targets regular cases rather than the worst-cases can perform better. The key challenge is capturing the "regular cases." A positive and tunable parameter (e.g., the parameter $\beta$ in the above example) is introduced in this paper to characterize the likelihood of an input. An online scheduling algorithm that can incorporate the prediction information with a tunable occurrence probability is proposed to achieve a good performance under "regular case" inputs.

## III. PROBLEM STATEMENT

In this section, we provide the general model for the online Deadline-constrained Energy-efficient Flow Scheduling (DEFS) problem in the data center networks. We first describe the model of the data center flow scheduling and link energy consumption. Then, we provide the model for flow prediction. Finally, we formulate the DEFS problem.

### A. Data Center Flow Scheduling

In this paper, the data center network is abstracted as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes (i.e., switches and hosts) and $\mathcal{E}$ is the set of network links. Typically, nodes are connected by links according to regular architectures like Fat-Tree (Figure 2(a)) or DCell (Figure 2(b)). We assume that all nodes and links in the network are identical. This is reasonable since the networks are built with identical commodity switches in modern data centers. Each link in the network is associated with an egress buffer. When a packet has been processed by the switch, it will be pushed into the egress buffer and wait to be transmitted to the next hop. Both the transmission order and the transmission rate are determined by the link according to some preset flow scheduling policy.



(a) FatTree (b) DCell

Fig. 2. Two commonly used date center network toplogies.

Assume that time is slotted in this system. Let $T$ denote the time-horizon considered. Then, an arbitrary time slot is indexed by an integer in time slot set $\mathcal{T} = \{0, 1, \cdots, T\}$. For each time slot, $t \in \mathcal{T}$, there is a set of deadline-constrained flows, $\mathcal{J}_e(t) = \{j_1, j_2, \cdots, j_n\}$, that have to be transmitted over link, $e \in \mathcal{E}$, from a source node to a destination node. We assume that the path of each flow can be computed as soon as the flow has been inserted into the network according to some regular routing algorithms in data center networks (such as ECMP). We also make the assumption that once the transmission begins, the route of each flow will never change. Let $\mathcal{J}(t) = \{\mathcal{J}_e(t) | e \in \mathcal{E}\}$ represent all the flows that remain in the network at time slot $t$ (including both the flows released at time slot $t$ and the flows released before time slot $t$ that have not been transmitted yet). Let $\mathcal{J} = \bigcup_{t \in \mathcal{T}} \mathcal{J}(t)$ represent the set of all the flows in the network. Each flow, $j_i \in \mathcal{J}$, is associated with four parameters, listed as follows.

a) the flow demand, which is the amount of data that needs to be routed (i.e., $w_i$),

b) the release time (i.e., $r_i$) and deadline (i.e., $d_i$),

c) the source (i.e., $p_i$) and destination (i.e., $q_i$), and

d) the path from the source to the destination (i.e., $\mathcal{P}_i$).

Assuming that preemption is allowed, a schedule is defined as a set

$$\mathcal{S} = \{(s_i(t), \mathcal{P}_i) | \forall j_i \in \mathcal{J}, \forall t \in \mathcal{T}\}. \quad (1)$$

where $s_i(t) \in \mathcal{R}$ is the transmission rate chosen for flow $j_i$ at time $t$ and $\mathcal{P}_i$ is the set of links that are on the chosen path for carrying the traffic from this flow. The transmission rate of each flow is a real number. Therefore, there is always a schedule that satisfies the feasibility, defined as follows.

**Definition 1.** *A schedule, $\mathcal{S}$, is called "feasible" if every flow can be accomplished within its deadline following this schedule.*

### B. Energy Consumption

We consider using speed scaling as an architectural support for network conservation. The key principle is that network devices can be designed so that slower operation speeds use a lower power supply [21]. We adopt the energy consumption function from [7] that has been widely used in existing literature. This model assumes that the energy consumption

behaviors of all the controllable network components can be abstracted as functions of the transmission rate of its associated link. For each link, $e \in \mathcal{E}$, a power consumption function, $g_e(x_e)$, is given to characterize the manner in which energy is being consumed with respect to the transmission rate, $x_e$, of link $e$. Since we assume that all the links in the network are identical, we adopt a uniform energy function for all the links in the network. Formally, for every link we are given a function $g(\cdot)$ which is expressed by

$$g(x_e) = \sigma + \mu x_e^{\alpha}, \qquad 0 \leq x_e \leq C, \tag{2}$$

where $\sigma$, $\mu$ and $\alpha$ are constants. The exact values of these parameters depend on the details of the technology. $C$ is the maximum transmission rate of a link. Normally, we have $\alpha > 1$, thus the power function $g(\cdot)$ is superlinear. To minimize the energy consumption, a feasible schedule will choose a transmission rate that is as small as possible but can still process each flow before its deadline. Thus, the constraint $0 \leq x_e \leq C$ can be relaxed and omitted in this paper.

### C. Model for Prediction and Uncertainty

Now, we present the probability-based model adopted in this paper. Assume that time is slotted, and index a time-slot by an integer in $\mathcal{T} = \{0, \cdots, T\}$, where $T$ is the time-horizon considered. Let $w_{t_0,t_1}$ denote the total flow demands associated with arrival time $t_0$ and deadline $t_1$. In practice, there are considerable uncertainties for the flow demands in data center networks. We define the $(T-t+1) \times 1$ vector $\mathbf{w}(t) = [w_{t,t}, \cdots, w_{t,T}]$ to denote the total flow demands at arrival time $t$. Note that the scheduler knows the precise value of $\mathbf{w}(t)$ only at and after time-slot $t$ because the value of $\mathbf{w}(t)$ is only revealed at time $t$. At time $s < t$, the value of $\mathbf{w}(t)$ is uncertain to the scheduler. However, the scheduler can use various sources of information to predict the future values of these uncertain quantities and improve its decision.

The probability-based prediction model predicts the probability distribution of each future flow demand, $\mathbf{w}(t)$. Assume that a prediction should be made $L > 0$ time slots ahead. That is, at time $t_p = \max\{0, (t-L)\}, t \in \mathcal{T}$, a prediction is available for every $t \in \mathcal{T}$. Specifically, for each $t \leq t_1 \leq T$ in time slot $t_p$, the probability-based prediction model predicts the probability distribution of $w_{t,t_1}$. Let $\widehat{f}_{t_p,t,t_1}(w)$ denote the predicted probability distribution of $w_{t,t_1}$ made in time slot $t_p$; the variable $w_{t,t_1}$ follows:

$$w_{t,t_1} \sim \widehat{f}_{t_p,t,t_1}(w). \tag{3}$$

There are various methods that can be used to predict the probability distribution of data flows in data center networks [20]. For each time slot $t$, the probability that the value of the total flow demand equals a possible value $w_0$ with an arriving time $t$ and a deadline $t_1$ is predicted to be $\widehat{f}_{t_p,t,t_1}(w_0)$ at time $t_p$. Thus, a prediction provides $(T-t+1) \times 1$ vectors $\widehat{\mathbf{f}}(t_p, t)$, representing the predicted probability distribution of the flow demands in $T-t+1$ time slots. Obviously, the prediction for all $t$ will not be known until time slot $t = \max\{t_1 - L, 0\}$.

With the above probability-based model, we can formulate the information set revealed at time $t$. Specifically, We only have interest in possible realizations that have a high probability according to the prediction. This is reasonable since, in reality, people more often face common cases rather than rare cases. Let $\widehat{f}^u_{t_p,t,t_1}$ denote the upper bound of $\widehat{f}_{t_p,t,t_1}(w)$, i.e., $\widehat{f}^u_{t_p,t,t_1} \geq \widehat{f}_{t_p,t,t_1}(w)$. Let $\beta$ denote a tunable parameter. The value of the parameter $\beta$ satisfies $0 \leq \beta \leq \min_{0 \leq t \leq t_1 \leq \mathcal{T}} \widehat{f}^u_{t_p,t,t_1}$. Thus, for each $\beta$ at time slot $t$, we will have a prediction vector $\widehat{\mathbf{w}}_\beta(t_p, t)$ for $\mathbf{w}(t)$ that satisfies the following:

$$\widehat{\mathbf{w}}_\beta(t_p, t) = \{\widehat{w}^\beta_{t_p,t,t'} | t' \in [t, T]\}, \tag{4}$$

$$\widehat{w}^\beta_{t_p,t,t'} \in \{w | \widehat{f}_{t_p,t,t_1}(w) \geq \beta\}. \tag{5}$$

We can now formulate the information set revealed at time slot $t$. Let $Z^\beta_t$ denote the information set revealed at time slot $t$, i.e., $Z^\beta_t = \{\mathbf{w}(t') | t' \in [0, t]\} \bigcup \{\widehat{\mathbf{w}}_\beta(t'_p, t') | t' \in [0, \min\{t + L, \mathcal{T}\}]\}$. In other words, the information revealed at time slot $t$ involves all flow demands that have already arrived (the first part) and all predictions that have a probability higher than $\beta$ (the second part).

### D. Problem Formulation

Assume that the routing protocol is given. We define $\mathcal{E}_a$ as the set of links that are used to carry flows according to the given routing protocol. Consequently, the total energy consumed by all links during $\mathcal{T}$ in a schedule $\mathcal{S}$ can be expressed by

$$\Phi_f(\mathcal{S}) = T \cdot |\mathcal{E}_a| \cdot \sigma + \int_0^T \sum_{e \in \mathcal{E}_a} \mu(x_e(t))^\alpha dt. \tag{6}$$

where $x_e(t)$ is the transmission rate of link $e$ at time $t$. The objective of the Deadline-constrained Energy-efficient Flow Scheduling (DEFS) problem is to find a feasible schedule, $\mathcal{S}$, such that the total energy consumption, $\Phi_f(\mathcal{S})$, is minimized. Since the paths are prefixed, the number of active links (i.e., $|\mathcal{E}_a|$) is a constant. Thus, the optimization objective can be simplified to $\Phi_f(\mathcal{S}) = \int_0^T \sum_{e \in \mathcal{E}_a} \mu(x_e(t))^\alpha dt$.

Assume that all the future flow arrival information is known *a priori*. The DEFS problem can be solved optimally in polynomial time by applying the Ellipsoid Algorithm or using the efficient combinatorial algorithm proposed by Lin *et. al.* in [11]. However, the assumption that all future information is known *a priori* in the offline setting is impractical in the real world. The information of flows is revealed over time. Therefore, we are interested in designing online algorithms that schedule a network flow demand that minimizes the total energy consumption in the network.

At each time $t \in \mathcal{T}$, an online algorithm, $A$, must determine the amount of energy used to transmit the flows by specifying the transmission rates for each flow based only on the knowledge of $Z_t$. In other words, the decision at time $t$ cannot be based on the values of any quantity that will be revealed in the future. Assuming that $Z$ is a possible realization of flow information, we use $\Phi^*_o(Z)$ to represent the optimal offline

**Algorithm 1** : **DEFSA**

**Input:** Time-slot $t$, $Z_t$, $\beta$.

1: **for** $e \in \mathcal{E}$ **do**
2: **for** $j_i \in \mathcal{J}_e(t)$ **do**
3: Calculate $\mathcal{P}_{e \rightsquigarrow q_i}$.
4: $w_i \leftarrow |\mathcal{P}_{e \rightsquigarrow q_i}| \cdot w_i$.
5: **end for**
6: Calculate the value of $\sigma_e(t)$.
7: Calculate $x_e(t)$ on set $\mathcal{J}_e(t)$.
8: Transmit flows by the EDF policy using $\sigma_e(t)x_e(t)$ as the transmission rate for subflow $j_i$.
9: **end for**

solution to DEFS under the realization $Z$. Let $\Phi_A(Z)$ be the solution of any online algorithm $A$. Clearly, we must have $\Phi_A(Z) \geq \Phi_o^*(Z)$. We can then evaluate the effectiveness of an online algorithm, $A$, by comparing it to the optimal solution of the offline algorithms mentioned above. Let $\mathcal{Z}$ denote the set of all possible realizations. The competitive ratio is defined as follows:

**Definition 2.** *Let $\Phi_o^*$ be the optimal offline solution to the problem. Let $\Phi_A$ be the solution of an online algorithm, $A$. Then, the competitive ratio of $A$ is $\sigma$ ($\sigma = \max_{Z \in \mathcal{Z}}\{\frac{\Phi_A(Z)}{\Phi_o^*(Z)}\}$), if for all sequences of input $\mathcal{Z}$.*

## IV. An Efficient online algorithm for the DEFS problem

We present an online algorithm that can incorporate future flow prediction information for the DEFS problem (referred to as DEFSA) in this section. DEFSA is an efficient heuristic algorithm that is designed based on our prediction model. The overview of the DEFSA algorithm executed at each time slot is illustrated in Algorithm 1.

When a flow, $j_i \in \mathcal{J}_e(t)$, is arriving on an edge, $e \in \mathcal{E}$, the scheduler calculates the remaining routing path, $\mathcal{P}_{e \rightsquigarrow q_i}$, from its current location to destination $q_i$ for this flow (in line 3) according to some given rules (such as the shortest path routing protocol) first. Then, the scheduler updates the *effective demand* by multiplying $w_i$ by the length of the remaining path, $\mathcal{P}_{e \rightsquigarrow q_i}$, in line 4. The intuition is that the execution of the flow $j_i$ must be complete by the deadline $d_i$ along its routing path. When all the preliminaries are met, the scheduler starts to compute the transmission rate as well as the transmission order for each flow.

The key insight of this algorithm is to intuitively compute the lowest possible rate using the optimal offline schedule algorithm (to simplify the description, we refer to this algorithm as YDS), which is raised by Yao *et. al.* [22] to solve the rate scaling problem in a single processor, given the tasks that have arrived to-date, and run at several times (a positive value) that speed. Note that the positive value represents a scaling factor. Specifically, for each link, $e$, the scheduler will behave as follows: At each time slot, $t$, the scheduler firstly calculates the value of the scaling factor which is denoted by $\sigma_e(t)$ (in line 6). Then, it uses the YDS algorithm to compute the optimal peak transmission rate, $x_e(t)$ on flow set, $\mathcal{J}_e(t)$, (assuming that no flows arrives after time slot $t$) from a global perspective (in line 7). After that, each flow, $j_{i_e} \in \mathcal{J}_e(t)$, is transmitted according to the Earliest-Deadline-First (EDF) [23] policy using $\sigma_e(t)x_e(t)$ as the transmission rate (in line 8). Note that the EDF policy is adopted here in order to meet the deadline constraint as much as possible.

It is not difficult to see that the value of $\sigma_e(t)$ plays a key role in determining the performance of the algorithm. In the following, we will show how to calculate the value of $\sigma_e(t)$ so that it can achieve an improved performance of the online algorithm by integrating the prediction knowledge of the future information into the computation.

Computation of the factor $\sigma_e(t)$ is inspired by the fundamental computation framework introduced in [24] which aims to smooth the peak electricity provision for vehicle recharge problems. We extend the computation framework so that it can incorporate the probability-based prediction model and meet the requirement of the data center network's energy consumption reduction scenario in this paper. Before going into details, we first introduce the following lemma.

**Lemma 1.** *([24]) An online algorithm, $A$, is feasible iff for all $Z \in \mathcal{Z}$ and all $t_1 \leq t_2$, $t_1, t_2 \in \mathcal{T}$, the following inequality holds:*

$$\sum_{t=t_1}^{t_2} \sum_{s=t}^{t_2} w_{t,s} \leq \int_{t_1}^{t_2} x_e(t)dt. \tag{7}$$

Lemma 1 states that, in order for an online algorithm to be feasible, the total traffic that can be transmitted according to the transmission rate chosen by the scheduler at any time interval, $[t_1, t_2]$, must be no smaller than the total number of flow demands that must be transmitted in the same interval. Further, Lemma 1 is also a sufficient condition. Specifically, if an online algorithm, $A$, satisfies Lemma 1, and the algorithm uses the Earliest-Deadline-First (EDF) policy [23] to transmit the flow, then the algorithm can transmit all the flow demands before their corresponding deadlines. The detailed proof of Lemma 1 has been presented in [24], and is omitted here due to space limitations.

Due to the superlinear property of the power function, a schedule that minimizes the overall energy consumption will try to use the smallest possible transmission rates to process the delayable flows throughout the time interval. For each time interval, $[t_1, t_2] \in \mathcal{T}$, we can find a positive value, $\sigma_{t_1,t_2}$, satisfying $x_e(t) \leq \sigma_{t_1,t_2} x_e^*(t)$, where $x_e^*(t)$ is the optimal transmission rate at time slot $t \in [t_1, t_2]$ and is computed using the YDS algorithm with the whole input sequence.

Combining $x_e(t) \leq \sigma_{t_1,t_2} x_e^*(t)$ with Lemma 1, we have the following:

$$\sum_{t=t_1}^{t_2} \sum_{s=t}^{t_2} w_{t,s} \leq \int_{t_1}^{t_2} x_e(t)dt \leq \sigma_{t_1,t_2} \int_{t_1}^{t_2} x_e^*(t). \tag{8}$$

Then, we have

$$\sigma_{t_1,t_2} \geq \frac{\sum_{t=t_1}^{t_2}\sum_{s=t}^{t_2} w_{t,s}}{\int_{t_1}^{t_2} x_e^*(t)}. \quad (9)$$

Unfortunately, one cannot obtain the value of $x_e^*(t)$ since we cannot know the whole input before the scheduling begins. However, considering a time slot $t \in [t_1, t_2]$, we know all the flows that arrive during $t' \in [t_1, t]$ and all the flow predictions for future time intervals, $[t, \min\{t+L, t_2\}]$. Thus, we are able to estimate an infimum for the value of $x_e^*(t)$ based on the information that is already known.

Let $\mathcal{Z}_\beta$ denote the subset of all possible realizations with a probability no smaller than $\beta$, i.e., $\mathcal{Z}_\beta = \{Z_t^\beta | t \in \mathcal{T}, Z_t^\beta$ satisfies the probability-based prediction model with $\beta\}$. Using the flow information mentioned above and assuming that no flows arrive after time slot $t_2$, we can estimate an infimum of the transmission rate for each time slot $t \in [t_1, t_2]$ (denoted by $x_e^*(Z_t^\beta)$). The estimation process is as follows. Given a realization $Z'(Z' \in \mathcal{Z}_\beta, Z'_t = Z_t^\beta)$, we can compute the optimal solution using YDS. Let $x'_e(Z_t^\beta)$ denote the transmission rate chosen by the optimal solution. Then, define the minimum value of $x'_e(Z_t^\beta)$ amongst all possible realizations in $\mathcal{Z}_\beta$ as $x_e^*(Z_t^\beta)$

$$x_e^*(Z_t^\beta) = \inf_{Z' \in \mathcal{Z}_\beta, Z'_t = Z_t^\beta} x'_e(Z_t^\beta). \quad (10)$$

Obviously, the value of $x_e^*(Z_t^\beta)$ provides an infimum for the link transmission rate under the flow information, revealed as $Z_t^\beta$. Thus, the following inequation is established:

$$x_e^*(Z_t^\beta) \leq x_e^*(t). \quad (11)$$

Substituting $x_e^*(t)$ with $x_e^*(Z_t^\beta) \leq x_e^*(t)$ for inequation (12),

$$\sigma_{t_1,t_2} \geq \frac{\sum_{t=t_1}^{t_2}\sum_{s=t}^{t_2} w_{t,s}}{\int_{t_1}^{t_2} x_e^*(Z_t^\beta)}. \quad (12)$$

By solving the following optimization problem,

$$\sigma_{t_1,t_2} = \sup_{Z \in \mathcal{Z}_\beta} \frac{\sum_{t=t_1}^{t_2}\sum_{s=t}^{t_2} w_{t,s}}{\int_{t_1}^{t_2} x_e^*(Z_t^\beta)} \quad (13)$$

and finding the largest $\sigma_{t_1,t_2}$ among all $t_1 < t_2, t_1, t_2 \in \mathcal{T}$,

$$\sigma_e(t) = \max_{t_1 \leq t_2, t_1, t_2 \in \mathcal{T}} \{\sigma_{t_1,t_2}\} \quad (14)$$

the factor $\sigma_e(t)$ has been computed. Note that the problem (13) is a bi-level optimization problem. Since the flow demands can be scaled up and down freely without violating the constraints of the system, the problem (13) can be converted to a convex problem, thus can be solved efficiently. [19]

Note that the value of $\beta$ represents a trade-off between conservative and aggressive behaviors. If the value of $\beta$ is set to 0, the algorithm exhibits a conservative behavior with a pessimistic vision of the future. Thus, it always uses a larger rate to transmit flows than the optimal solution does; If the value of $\beta$ is set to a large nonzero value, the algorithm exhibits an aggressive behavior with an optimistic vision of the future. Thus, we always use a smaller, but also feasible, rate to transmit flows rather than the optimal solution. The optimal value of $\beta$ depends on the actual flow behavior of the applications in a specific data center network. In practice, a desirable value of $\beta$ can be found by conducting several experiments via the network manager.

## V. Numerical Results

In this section, we briefly describe numerical results that illustrate the performance of the proposed online algorithm.

### A. Settings

**Data center.** We use a 8-ary Fat-Tree topology (Figure 2(a) to shows a 4-ary Fat-Tree topology) simulate a data center network in this evaluation. The 8-ary Fat-Tree topology consists of 8 pods, each involving 16 edge switches. In total, 128 servers are connected by these edge switches.

**Benchmarks.** To evaluate the energy-saving performance of our online algorithm, we compare the evaluation results with the Most-Critical-First (referred to as MCF for brevity) algorithm, which is an offline scheduling algorithm that can obtain the optimal solution. [11] Even though determining the optimal solution in a real network is not available, the optimal solution can serve as the lower bound. In addition to the benchmarking of the optimal solution, we also compare the DEFSA algorithm with another famous online flow scheduling algorithm (referred to as BKP for brevity) [17] that schedules flows without prediction information.

**Parameters.** The energy function used in this evaluation is $x^\alpha$. We choose 2, 3, and 4 as the values of $\alpha$ to see the effect on the results. We set the time-horizon to $[1, 100]$. The arrival-flow sequence is generated in the following way. The number of flows varies from 100 to 350. The release times and deadlines of each flow are uniformly distributed from $[1, 100]$. The demand of each flow is a random value that follows the normal distribution $\mathcal{N}(10, 3)$. Each prediction is made 30 time slots ahead and follows a normal distribution.

### B. Results

**Measuring the impact of incorporating prediction.** We evaluate the impact of incorporating the prediction information in the scheduling. In order to understand the influence of the value of $\beta$ on the performance of the DEFSA algorithm, we set 0.1, 0.3 and 0.7 as the values of $\beta$, which are referred to as DEFSA-I, DEFSA-II, and DEFSA-III, respectively. The simulation results are shown in Table I. The results illustrate the amount of energy consumed in the network with three different flow scheduling strategies when the number of flows is set to 100 and 350, respectively. The value of $\alpha$ is set to 4. As expected, the offline optimal algorithm outperforms two online algorithms to a large extent. Besides, the DEFSA algorithm performs much better than the BKP algorithm. Therefore, by incorporating future predictions into the scheduling, we can significantly reduce the energy consumption in data center networks much further.

In addition, Table I also shows the impact of different values of $\beta$ on the performance of the DEFSA algorithm. It demonstrates that the DEFSA algorithm performs better in the setting with $\beta = 0.3$ (i.e., DEFSA-II) than the other two settings (i.e., $\beta = 0.1$ in DEFSA-I and $\beta = 0.7$ in DEFSA-III). It shows that choosing different values for $\beta$ results in different energy consumption reductions. This confirms that there is a trade-off between conservative and aggressive behaviors in terms of energy conservation. Neither an oversized (aggressive) nor undersized (conservative) value for $\beta$ is good for improving the energy efficiency of the data center network. Since the quality of the value of $\beta$ is closely related to the network scene, a desirable value of $\beta$ can be determined by repeated experiments.

TABLE I
THE AMOUNT OF ENERGY CONSUMED IN THE NETWORK PROVIDING THREE DIFFERENT FLOW SCHEDULING STRATEGIES.($\times 10^5$)

| Case | MCF | DEFSA-I | DEFSA-II | DEFSA-III | BKP |
|---|---|---|---|---|---|
| 100# | 3 | 91 | 63 | 90 | 336 |
| 350# | 10171 | 405494 | 205976 | 325494 | 1086340 |

**Measuring the impact of the value of $\alpha$.** We then evaluate the impact of the value of $\alpha$ on the performance of the proposed algorithm. We set the values of $\alpha$ as 2, 3, and 4. The flow number is set to 100. The simulation results are shown in Figure 3. From the results, it can be seen that all three test algorithms consume more energy with a larger value of $\alpha$. Besides, with different settings of the value of $\alpha$, we find that the BKP algorithm still uses more energy than the DEFSA algorithm.

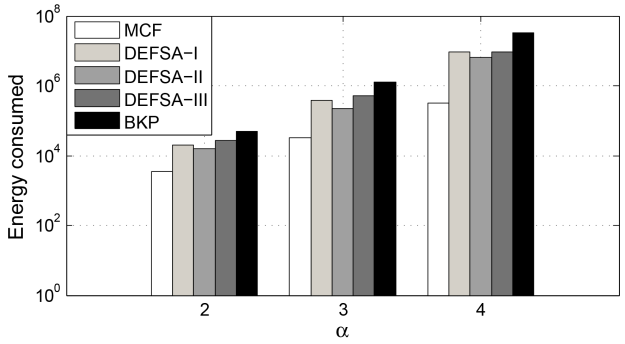



Fig. 3. The amount of energy consumed in the network providing three different values of $\alpha$ under different flow scheduling strategies. (100 flows)

**Measuring the impact of the value of the lookahead time interval.** We then evaluate the impact of the value of the lookahead time interval $L$ on the performance of the proposed algorithm. The value of $\alpha$ is set to 4. The value of $\beta$ is set to 0.3. The flow numbers are set to 100 and 350, respectively. We set the values of $L$ as 0, 30 and 60, which are referred to as DEFSA-L0, DEFSA-L30, and DEFSA-L60, respectively. The simulation results are shown in Figure II. From the results, it can be seen that, when $L$ is set to 0, which means that no future information is provided, the energy consumption is the highest in both flow scenarios. As the value of $L$ increases, more future information is known, and the performance of the DEFSA algorithm is improved. These results also demonstrate that incorporating future predictive information is beneficial to improving the performance of the scheduling.

TABLE II
THE AMOUNT OF ENERGY CONSUMED IN THE NETWORK PROVIDING THREE DIFFERENT FLOW SCHEDULING STRATEGIES.($\times 10^5$)

| Case | DEFSA-L0 | DEFSA-L30 | DEFSA-L60 |
|---|---|---|---|
| 100# | 282 | 63 | 23 |
| 350# | 995307 | 205976 | 91342 |

**Measuring the impact of the number of flows.** We also evaluate the impact of the number of flows on the performance of the proposed algorithm. The simulation results are shown in Figure 4. The results show that the DEFSA algorithm outperforms the BKP algorithm in variable traffic scenarios.

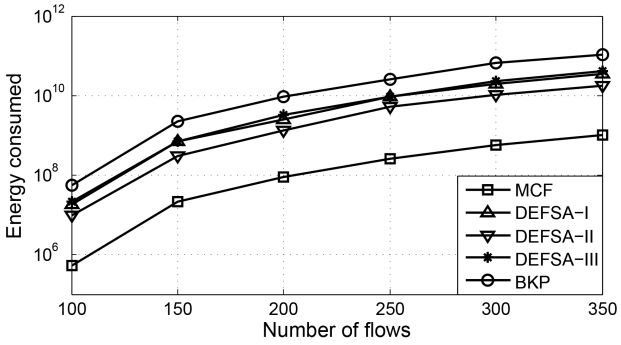



Fig. 4. The amount of energy consumed in the network varies with the number of flows under different flow scheduling strategies.

## VI. RELATED WORK

This section summarizes the existing studies on energy-efficient flow scheduling that are most related to this paper.

### A. Offline Scheduling

The pioneering work on saving energy in computer systems via task scheduling based on speed scaling is by Yao *et. al.* [22]. They propose an optimal, offline, energy-efficient task scheduling algorithm for a single processor. Inspired by that work, Li *et. al.* [25] propose a preemptive scheduling method to reduce the energy consumed in data center networks. Their method can remove bottleneck active links in data centers, thus improving the overall utilization of network resources. The authors in [26] propose an energy-efficient flow scheduling in software defined networks. This method minimizes the total energy consumption in data center networks while ensuring that each flow can be transmitted before the deadline. In [27], the authors propose a network flow scheduling algorithm that can achieve a trade-off between the delay, queue size, and energy conservation. There are also a number of studies [7, 11, 28, 29] that focus on the topic of using both routing and scheduling to reduce energy consumption in wired networks.

### B. Online Scheduling

There are few works that target energy-efficient online scheduling the way that we do. Yao *et. al.* propose two online scheduling heuristics for a single processor in [22]. The first is called Average Rate (AVR), which runs each job with a rate equal to its density. The second is called Optimal Available (OA), which runs each job with a rate that would be optimal according to its current jobs. A more competitive scheduling is proposed by Bansal *et. al.* in [17] with a competitive ratio no larger than $2(\frac{\alpha}{\alpha-1})^{\alpha}e^{\alpha}$. The authors in [18] consider online packet scheduling with hard deadlines and weights through given routes in a general multihop wired network. They study packet scheduling as well as admission control to maximize the cumulative weights of packets that have been successfully transmitted within the deadline. However, they neither consider energy conservation nor incorporate future information. The computation process of the tunable parameter in our paper shares some similarities to the technique introduced by Zhao *et. al.* in [24]. However, there are two main differences exist between these works: 1) while [24] focuses on minimizing the peak resource requirement from the grid, our work aims to find a schedule that minimizes the total energy consumption of a network; 2) while [24] only computes the competitive ratio that satisfies the worst-case performance, our method can achieve an arbitrary trade-off between conservative and aggressive behaviors in terms of energy consumption.

## VII. Conclusion

This paper has tackled the online flow scheduling problem under the deadline constraint for energy conservation in data center networks. We propose a probability-based flow prediction model that captures the uncertainty of the data center network system. To better exploit the available future knowledge obtained by the predictions, we develop a tunable and efficient online flow scheduling algorithm to reduce the energy consumption while guaranteeing flow deadlines in data center networks based on the probability-based model. By introducing a scaling factor on bandwidth allocation, the online algorithm allows the network manager to obtain an arbitrary trade-off between conservative and aggressive behaviors in terms of power conservation. We demonstrate the effectiveness of the algorithm through both theoretical analysis as well as numerical evaluations.

## References


[1] J. Glanz, "Power, pollution and the internet," *The New York Times*, 2012.

[2] J. Koomey, "Growth in data center electricity use 2005 to 2010," *A report by Analytical Press, completed at the request of The New York Times*, vol. 9, 2011.

[3] D. Anderson, T. Cader, T. Darby, N. Gruendler, R. Hariharan, A. Holler, C. Lindberg, C. Long, P. Morris, A. Rawson *et al.*, "A framework for data center energy productivity," *White Paper*, vol. 13, 2008.

[4] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, pp. 732–794, 2016.

[5] C. Gunaratne, K. J. Christensen, B. Nordman, and S. Suen, "Reducing the energy consumption of ethernet with adaptive link rate (ALR)," *IEEE Trans. Computers*, vol. 57, no. 4, pp. 448–461, 2008.

[6] P. Patel-Predd, "Update: Energy-efficient ethernet," *IEEE Spectrum*, vol. 45, no. 5, 2008.

[7] M. Andrews, A. F. Anta, L. Zhang, and W. Zhao, "Routing for power minimization in the speed scaling model," *IEEE/ACM Trans. Netw.*, vol. 20, no. 1, pp. 285–294, 2012.

[8] B. Zhou, F. Zhang, L. Wang, C. Hou, A. F. Anta, A. V. Vasilakos, Y. Wang, J. Wu, and Z. Liu, "HDEER: A distributed routing scheme for energy-efficient networking," *IEEE JSAC*, vol. 34, no. 5, pp. 1713–1727, 2016.

[9] L. Wang, F. Zhang, J. A. Aroca, A. V. Vasilakos, K. Zheng, C. Hou, D. Li, and Z. Liu, "Greendcn: A general framework for achieving energy efficiency in data center networks," *IEEE JSAC*, vol. 32, no. 1, pp. 4–15, 2014.

[10] L. Wang, F. Zhang, C. Hou, J. A. Aroca, and Z. Liu, "Incorporating rate adaptation into green networking for future data centers," in *IEEE NCA, Cambridge, MA, USA, August 22-24,*, 2013, pp. 106–109.

[11] L. Wang, F. Zhang, K. Zheng, A. V. Vasilakos, S. Ren, and Z. Liu, "Energy-efficient flow scheduling and routing with hard deadlines in data center networks," in *IEEE ICDCS, Madrid, Spain, June 30 - July 3*, 2014, pp. 248–257.

[12] M. Andrews, A. F. Anta, L. Zhang, and W. Zhao, "Routing and scheduling for energy and delay minimization in the powerdown model," *Networks*, vol. 61, no. 3, pp. 226–237, 2013.

[13] H. Chan, W. Chan, T. W. Lam, L. Lee, K. Mak, and P. W. H. Wong, "Energy efficient online deadline scheduling," in *ACM-SIAM SODA, New Orleans, Louisiana, USA, January 7-9*, 2007, pp. 795–804.

[14] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *ISCA, June 19-23, Saint-Malo, France*, 2010, pp. 338–347.

[15] L. Huang, Q. Jia, X. Wang, S. Yang, and B. Li, "Pcube: Improving power efficiency in data center networks," in *IEEE CLOUD, Washington, DC, USA, 4-9 July*, 2011, pp. 65–72.

[16] C. Wilson, H. Ballani, T. Karagiannis, and A. I. T. Rowstron, "Better never than late: meeting deadlines in datacenter networks," in *ACM SIGCOMM, Toronto, ON, Canada, August 15-19*, 2011, pp. 50–61.

[17] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed scaling to manage energy and temperature," *J. ACM*, vol. 54, no. 1, pp. 3:1–3:39, 2007.

[18] Z. Mao, C. E. Koksal, and N. B. Shroff, "Online packet scheduling with hard deadlines in multihop communication networks," in *IEEE INFOCOM, Turin, Italy, April 14-19,*, 2013, pp. 2463–2471.

[19] S. Zhao, X. Lin, and M. Chen, "Peak-minimizing online EV charging," in *ALLERTON, Allerton Park & Retreat Center, Monticello, IL, USA, October 2-4*, 2013, pp. 46–53.

[20] D. Shen, J. Luo, F. Dong, and J. Zhang, "Appbag: Application-aware bandwidth allocation for virtual machines in cloud environment," in *IEEE ICPP, Philadelphia, PA, USA, August 16-19*, 2016, pp. 21–30.

[21] J. A. Aroca, A. Chatzipapas, A. F. Anta, and V. Mancuso, "A measurement-based analysis of the energy consumption of data center servers," in *e-Energy, Cambridge, United Kingdom - June 11 - 13*, 2014, pp. 63–74.

[22] F. F. Yao, A. J. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *FOCS, Milwaukee, Wisconsin, 23-25 October*, 1995, pp. 374–382.

[23] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[24] S. Zhao, X. Lin, and M. Chen, "Peak-minimizing online EV charging: Price-of-uncertainty and algorithm robustification," in *IEEE INFOCOM, Kowloon, Hong Kong, April 26 - May 1*, 2015, pp. 2335–2343.

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

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

[27] M. Andrews and L. Zhang, "Scheduling algorithms for optimizing the tradeoffs between delay, queue size and energy," in *CISS, Princeton, NJ, USA, March 21-23*, 2012, pp. 1–6.

[28] H. Zhu, X. Liao, C. de Laat, and P. Grosso, "Joint flow routing-scheduling for energy efficient software defined data center networks: A prototype of energy-aware network management platform," *J. Network and Computer Applications*, vol. 63, pp. 110–124, 2016.

[29] M. Andrews, S. Antonakopoulos, and L. Zhang, "Energy-aware scheduling algorithms for network stability," in *INFOCOM IEEE, 10-15 April, Shanghai, China*, 2011, pp. 1359–1367.