# EXR: Greening Data Center Network with Software Defined Exclusive Routing

Dan Li, Yunfei Shang, Wu He, and Congjie Chen

**Abstract**—The explosive expansion of data center sizes aggravates the power consumption and carbon footprint, which has restricted the sustainable growth of cloud services and seriously troubled data center operators. In recent years, plenty of advanced data center network architectures have been proposed. They usually employ richly-connected topologies and multi-path routing to provide high network capacity. Unfortunately, they also undergo inefficient network energy usage during the traffic valley time. To address the problem, many energy-aware flow scheduling algorithms are proposed recently, primarily considering how to aggregate traffic by flexibly choosing the routing paths, with flows fairly sharing the link bandwidths. In this paper, we leverage software defined network (SDN) technique and explore a new solution to energy-aware flow scheduling, i.e., scheduling flows in the time dimension and using exclusive routing (EXR) for each flow, i.e., a flow always exclusively utilizes the links of its routing path. The key insight is that exclusive occupation of link resources usually results in higher link utilization in high-radix data center networks, since each flow does not need to compete for the link bandwidths with others. When scheduling the flows, EXR leaves flexibility to operators to define the priorities of flows, e.g., based on flow size, flow deadline, etc. Extensive simulations and testbed experiments both show that EXR can effectively save network energy compared with the regular fair-sharing routing (FSR), and significantly reduce the average flow completion time if assigning higher scheduling priorities to smaller flows.

**Index Terms**—Data center network, software defined networking, exclusive routing

---

## 1 INTRODUCTION

THE explosive expansion of data center sizes aggravates the power consumption and carbon footprint. The statistics show that 1.5 percent of the global electricity usage came from data centers in 2011 [1], [2]. The ever increasing energy consumption of data centers has restricted the sustainable growth of cloud services and raised economic and environmental concerns. Recently people pay attention to the contribution of networking part to the energy consumed by the whole data center. In typical data centers, the network accounts for 10-20 percent of the overall power consumption [3], while the proportion can be up to 50 percent if the mature server-side power management techniques are employed [4].

In order to achieve higher network capacity, plenty of "richly-connected" data center network architectures have been proposed to replace the traditional tree architecture, such as Fat-Tree [5], BCube [6], FiConn [7], uFix [8], etc. These architectures employ abundant switches and links to enjoy full bisection bandwidth at traffic peak time. Although these architectures significantly enhance the network performance, there are two *side effects* on the data center energy consumption. First, they increase the energy consumption proportion of the networking part due to the usage of much more switches. Second, they cause inefficient network energy consumption at traffic valley time, because of the low utilization of switches. The fundamental reason for network

energy waste is that the energy consumption of today's switches is not proportional to the traffic loads [9], which leads to the fact that idle or under-utilized switches still consume considerable energy.

In order to save the network energy in "richly-connected" data centers, many energy-aware flow scheduling approaches are proposed [10], [11]. The general idea is to aggregate network flows into a subset of switches/links and use as few switches/links as possible to carry the flows. In this way, the active switches can carry more traffic and the idle switches can be put into sleep mode for energy saving. This kind of traffic aggregation naturally results in multiple flows sharing the same switches/links. Based on the TCP congestion control algorithm, the flows get their throughputs by fairly sharing the bandwidths of bottleneck links. We call this kind of energy-aware flow scheduling approach *fair-sharing routing*, or FSR for short.

In this work, we find that FSR in high-radix data center networks usually results in low utilization of some non-bottleneck switches/links, wasting the energy consumption on them. We thus design a more energy-efficient flow scheduling approach called *exclusive routing* (EXR for short), leveraging the emerging software defined network (SDN) technique. Instead of letting flows compete and fairly share the link bandwidths, EXR schedules flows in the time dimension and lets each flow always exclusively utilize the links of its routing path when transmitting data.

At any time, EXR always tries to spread flows to different paths, but it needs to guarantee that every link is occupied by at most one flow at the time, and there are no bottleneck active links in the network. If some flows have to inevitably share links in their routing paths, they are scheduled to transmit one after another without time overlap. EXR leaves flexibility to operators to assign flows

- *The authors are with the Computer Science Department, Tsinghua University, Beijing, China.*

with different scheduling priorities, and the priority can depend on flow size, flow deadline, etc. For example, if operators want to reduce the expected flow completion time, they can assign higher priorities to smaller-sized flows; while if the application has deadline requirements, flows with shorter deadlines should be assigned with higher priorities, so as to maximize the number of flows that can be completed before their deadlines.

EXR scheduling intelligence is put into the SDN controller and virtual switches inside the physical servers, and it requires no update of virtual machines or SDN switches. The design of EXR enjoys additional merits besides improving the energy efficiency of the network. First, it is unnecessary to synchronize the time among servers. The SDN controller globally manages the flows' states and explicitly notifies the virtual switches to permit or suspend a flow. Hence, the servers do not need to set timers to trigger flow transmission or worry about the time synchronization among each other. Second, there is no need of bandwidth allocation for specific flows. Since each active flow exclusively occupies all the links in its routing path, EXR does not require rate limiter at the sender or bandwidth reservation at switches for a flow.

We conduct extensive simulations to evaluate EXR in typical data center network topologies, including Fat-Tree and BCube. The results demonstrate that compared with FSR, EXR can effectively save network energy and reduce the average flow completion time if assigning smaller flows with higher scheduling priorities. Besides, we prototype EXR in an SDN environment, and the experiment on a small Fat-Tree testbed shows that EXR makes almost full utilization of the active links, which significantly outperforms FSR and helps reduce the network energy consumption.

## 2 BACKGROUND AND MOTIVATION

### 2.1 Energy Consumption Model of Data Center Networks

The power consumption of an active switch is composed of two parts, the *fixed* part and the *dynamic* part. The fixed part includes chassis, fans, switching fabric, etc., which always consumes constant power when the switch is on. The dynamic part includes the ports. A port can be put into sleep mode with almost zero power consumption, but consumes nearly full power in the active state. We take two examples, namely, Cisco Catalyst 6500 switch and Cisco Nexus 2224TP switch, to show the power consumptions of the two parts. The former switch contains two linecards and 48 Gigabit ports in each linecard. The power of the fixed part is 472 W while that of each port is 3 W [12]. Hence, the dynamic part occupies 38 percent of the switch's power. The latter switch contains one linecard and 24 Gigabit ports. Its fixed part consumes 48 W while each port consumes 2 W [13], which indicates that the dynamic part occupies 50 percent of the switch's power.

Since the energy consumption of a switch is not proportional to the traffic load [9] and the dynamic part contributes considerable power to the whole switch, a feasible approach to saving the energy of data center network is to let the idle switches/ports enter sleep mode. It is also recommended

by IEEE 802.3az standard [14]. We can then use Eq. (1) to calculate the total amount of network energy used for transmitting a group of traffic flows in a data center. $\mathcal{I}$ and $\mathcal{J}(i)$ denote the set of switches and the set of ports in switch $i$, respectively. $P_i$ denotes the fixed power consumption in switch $i$, $t_i$ denotes the active working duration of switch $i$, $Q_{i,j}$ denotes the power of port $j$ in switch $i$, and $t'_{i,j}$ denotes the active working duration of port $j$ in switch $i$.

$$E = \sum_{i \in \mathcal{I}} \left( P_i * t_i + \sum_{j \in \mathcal{J}(i)} Q_{i,j} * t'_{i,j} \right). \tag{1}$$

Since most data center networks employ a homogeneous set of switches to interconnect servers, we assume all the switches have equal fixed power $P$ and all the ports have equal power $Q$. We further suppose the bandwidth capacity of each port is $C$. Then we can calculate the total network energy consumption as Eq. (2), where $m_i$ and $m'_{i,j}$ denote the aggregate traffic amount traversing switch $i$ and its port $j$ respectively, while $U_i$ and $U'_{i,j}$ represent the average utilization ratio of switch $i$ and its port $j$ in the transmission duration, respectively,

$$E = P * \sum_{i \in \mathcal{I}} \frac{m_i}{U_i * C * |\mathcal{J}(i)|} + Q * \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}(i)} \frac{m'_{i,j}}{C * U'_{i,j}}. \tag{2}$$

From Eq. (2) we draw inspiration that energy consumption of a data center network can be effectively reduced by increasing the utilization ratios of active switches and ports/links, while putting idle switches/ports into sleep mode. Therefore, an energy-efficient flow scheduling approach should improve the utilization of active switches/links as high possible.

### 2.2 Switch/Link Utilization of Data Center Network

As aforementioned, existing works use FSR to schedule data center flows in an energy-efficient way. In order to motivate the necessity of introducing exclusive routing to improve the switch/link utilization, we make a measurement study on the utilization ratio of switches/links under FSR by simulation. We conduct the simulation in a Fat-Tree topology containing 24 pods (3,456 servers), and use ECMP [15] to choose the routing paths for network flows given the multiple equal-cost paths between servers. We randomly choose half servers and generate all-to-all traffic flows among them.

Figs. 1a and 1b show the cumulative distribution of the utilization of active switches and active links, respectively. It demonstrates that the switches/links with zero utilization are less than 5 percent. On the other side, most switches and links are active but have a low utilization. For switches, about 80 percent switches have a utilization ratio below 55 percent, and about 20 percent switches have a utilization ratio of less 40 percent. For links, the utilization of 80 percent links is no more than 65 percent, and about 24 percent links have less than 40 percent utilization. Only less than 1 percent switches and less than 10 percent links have a utilization higher than 90 percent. The low utilizations of active switches/links come from inevitable bandwidth competition among flows. These flows fairly share the
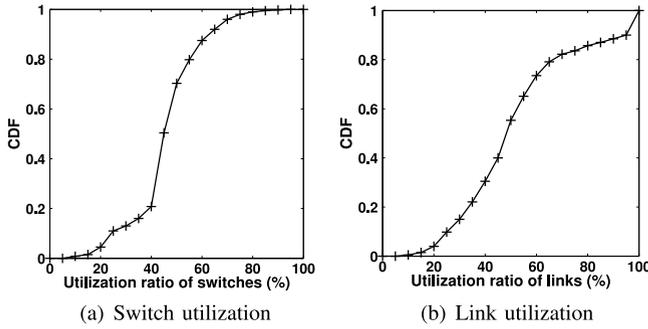
(a) Switch utilization        (b) Link utilization

Fig. 1. Cumulative distribution formulation (CDF) of active switch and link utilizations in a Fat-Tree network using FSR.



(a) Fair sharing routing        (c) Exclusive routing

Fig. 3. Scheduling sequence of the three flows. $t1 = \frac{m_1}{C}$, $t2 = \frac{2m_2}{C}$, $t3 = \frac{m_2+m_3}{C}$, $t1' = \frac{m_1}{C}$, $t2' = \frac{m_2}{C}$, $t3' = \frac{m_2+m_3}{C}$.

bandwidth of bottleneck links by TCP congestion control, and only fully utilize the bandwidth of the bottleneck links. Meanwhile, a great number of non-bottleneck links are under-utilized, which leaves room for more energy-efficient flow scheduling approaches.

## 3 EXR DESIGN

In this section, we present the design of EXR.

### 3.1 Working Example

In order to let readers get a clear idea of how EXR works and illustrate the benefit of EXR over FSR, we first give a working example as follows. As shown in Fig. 2, there is a partial 4-ary Fat-Tree network which contains 12 switches marked with "S" and eight servers marked with "H". We assume every switch has equal power consumption $P$, every port has equal power consumption $Q$, and every link has equal bandwidth capacity of $C$. Three flows need to be transmitted in the network, namely, flow 1 "H3 → H7", flow 2 "H1 → H5", and flow 3 "H1 → H8". The sizes of the three flows are $m1$, $m2$ and $m3$ respectively (without loss of generality, we assume $m1 \leq m2 \leq m3$). In the network, between any two inter-pod servers there are four available paths, traversing $S1$, $S2$, $S3$ and $S4$ respectively. We assume the three flows arrive at time $t = 0$ simultaneously. When scheduling the flows, flow 1 chooses the path across switch $S3$ (marked with yellow squares), while flow 2 selects the path across switch $S1$ (marked with red circles). The two flows can be transmitted immediately at $t = 0$ since they do not have overlapping links with each other. At what follows we discuss two scheduling solutions for flow 3.
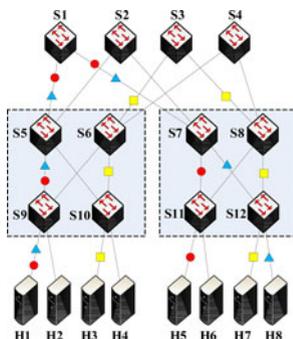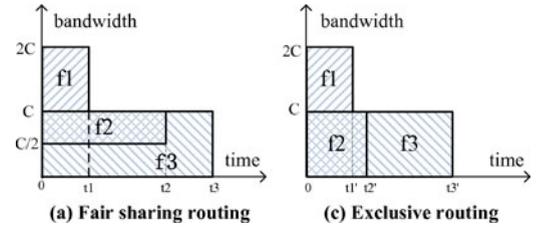
*FSR.* Flow 3 chooses the path across switch $S1$ (marked with blue triangles in Fig. 2), and it sends data at time $t = 0$, concurrently with flow 1 and flow 2.

*EXR.* Flow 3 chooses the same path as FSR, but it begins transmitting data only after flow 2 finishes at time $t = \frac{m_2}{C}$, since its path has overlapping links with flow 2.

We show the data transmission processes of the two solutions in Fig. 3, and show the flow completion time and network energy consumption in Table.1. When calculating the network energy, we only consider the active working duration of the involved switches, without taking the power state transition time into account. In FSR, flow 1 sends at rate $C$ as it exclusively occupies the routing path. From $t = 0$ to $t = \frac{2m_2}{C}$, flows 2 and 3 have overlapping links with each other, and hence the transmission rate for each is $\frac{C}{2}$. When flow 2 finishes, the remaining size of flow 3 ($m_3 - m_2$) can be sent at rate $C$. By calculating the number of active switches and ports used in each time period, FSR takes a total network power of $\frac{P}{C} * (5m_1 + 7m_2 + 5m_3) + \frac{Q}{C} * (10m_1 + 16m_2 + 10m_3)$ to finish transmitting the three flows.

If we use EXR, flow 3 is not sent before flow 2 finishes. It enables all the three flows to exclusively occupy the full bandwidth $C$ of all links in their routing paths during transmission. Since the bandwidth capacity of the corresponding links can be fully utilized, the overall network energy decreases to $\frac{P}{C} * (5m_1 + 5m_2 + 5m_3) + \frac{Q}{C} * (10m_1 + 10m_2 + 10m_3)$. We can easily figure out that EXR can indeed save network energy compared with FSR. In practical data center network with more switches/links and more flows, the room for energy saving will be much larger.

### 3.2 Design Overview

Fig. 4 shows the architecture overview of EXR. EXR works based on SDN. The EXR scheduling intelligence only requires modifying the SDN controller and the virtual switches within servers. The sender/receivers (virtual machines) of the flows



Fig. 2. Route selection of the three flows in a partial 4-ary Fat-Tree network.

TABLE 1
Flow Completion Time and Network Energy Consumption
in the Two Scheduling Solutions

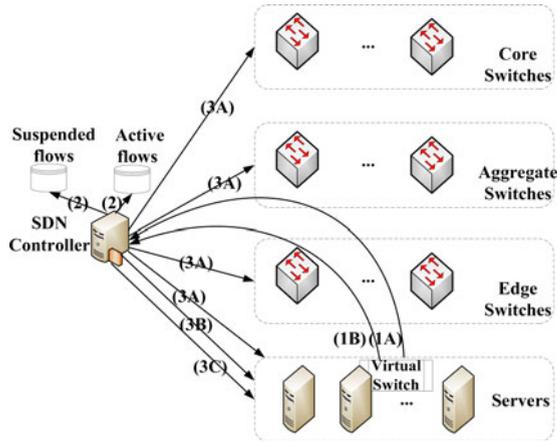|  | FSR | EXR |
|---|---|---|
| Flow 1's completion time | $\frac{m_1}{C}$ | $\frac{m_1}{C}$ |
| Flow 2's completion time | $\frac{2m_2}{C}$ | $\frac{m_2}{C}$ |
| Flow 3's completion time | $\frac{m_2+m_3}{C}$ | $\frac{m_2+m_3}{C}$ |
| Overall network energy | $\frac{P}{C} * (5m_1 + 7m_2 + 5m_3) + \frac{Q}{C} * (10m_1 + 16m_2 + 10m_3)$ | $\frac{P}{C} * (5m_1 + 5m_2 + 5m_3) + \frac{Q}{C} * (10m_1 + 10m_2 + 10m_3)$ |

Fig. 4. EXR design overview. (1A) The virtual switch receives a new flow and directs it to SDN controller. (1B) An active flow finishes and the virtual switch reports to SDN controller. (2) SDN controller runs EXR scheduling algorithm, and updates active flows and suspended flows. (3A) If a new flow becomes active, SDN controller installs forwarding entries in switches and notifies the virtual switch to permit the flow. (3B) If a suspended flow becomes active, SDN controller notifies the virtual switch to permit the flow. (3C) If a new flow or active flow becomes suspended, SDN controller notifies the virtual switch to block the flow.

and the physical switches in the network are both transparent to EXR. All the flows in the network are classified into two categories, active flows and suspended flow. The virtual switches at the senders permit active flows (at most one active flow from a sender at any time) but block suspended flows.

The EXR scheduling algorithm is triggered when a new flow arrives at the virtual switch or an active flow finishes transmission. In both cases, the virtual switch reports to the SDN controller, and the controller runs the scheduling algorithms presented in Section 3.3 to update the sets of active flows and suspended flows. There are three cases for the computation results. First, if a new flow becomes active, the controller immediately installs the forwarding entries in the switches and notifies the virtual switch at the sender to permit the flow. Second, if a suspended flow becomes active, the controller just notifies the virtual switch to permit the flow. Third, if an active flow becomes suspended, the controller notifies the virtual switch to block the flow. In whatever case, it is the controller that manages the global flow states. The servers do not need to maintain timers to launch specific flows, so it is unnecessary of time synchronization among servers for correct flow scheduling.

When the SDN controller schedules flows by EXR, every active flow exclusively occupies links in its routing path. As a result, there is no need of bandwidth reservation or rate limit for specific flows. If there is no flow in a link, the two ports at the ends of the link are put into sleep. When all the ports in a switch are asleep, the switch can be put into sleep. Network energy saving comes from high utilization of switches/links. EXR leaves the flexibility to operators to assign flows with different scheduling priorities, e.g., Shortest Job First (SJF) strategy or Earliest Deadline First (EDF) strategy. If shorter flows are assigned with higher priority, the waiting time of big flows should be considered to keep them from starvation. Note that in the current design, we simply assume all the flows are network-limited, i.e., the applications can generate traffic as fast as possible to saturate the bandwidth capacity of links. It is not difficult to

generalize EXR to support application-limited flows, by putting more flows in a link if the available bandwidth allows. But in order to present our idea in a clean way, throughout this paper we focus on network-limited flows.

### 3.3 Algorithms

Next we describe the EXR scheduling algorithms run at the SDN controller.

Algorithm 1 shows the overall scheduling algorithm of EXR. $F_{active}$ and $F_{suspend}$ denote the set of active flows and suspended flows, respectively. The scheduling is triggered when an active flow finishes transmission or a new flow arrives. The algorithm needs to decide both the routing path and the scheduling state (active or suspended) for each target flow. There are three cases. First, if a path without any other flows can be found for the target flow, the target flow becomes active. This path is called an *idle path* for the target flow. Second, there is no idle path for the target flow, but there exists a path in which all the existing flows in the path have lower priorities than the target flow. Then the target flow will become active, with existing flows in the path being preempted and suspended. This path is called a *preemptive path* for the target flow. Third, if there is neither idle path nor preemptive path for the target flow, the target flow will be suspended. In each case, if there are multiple eligible paths, the one that traverses the minimum number of idle switches will be preferentially chosen. The purpose is to increase the utilization ratio of active switches.

---

**Algorithm 1.** EXR Scheduling

**Input:**
$G$: data center network topology;
$F_{active}$: set of active flows;
$F_{suspend}$: set of suspended flows.
**Output:**
$\{<e.state, e.path>, \ \forall e \in F_{suspend} \cup F_{active}\}$: scheduling state and path of each flow.
1   **if** *a flow $f$ in $F_{active}$ finishes* **then**
2     $F_{active} \leftarrow F_{active} - \{f\}$;
3     **return ScheduleSuspend**($G$, $F_{active}$, $F_{suspend}$);
4   **end**
5   **else if** *a new flow $f$ arrives* **then**
6     $<t,p> \leftarrow$ **PathCalculation**($G$, $F_{active}$, $f$);
7     **if** $(t = IDLEPATH \parallel t = PREEMPTPATH)$ **then**
8       $f.state \leftarrow$ ACTIVE; $f.path \leftarrow p$;
9       $F_{active} \leftarrow F_{active} + \{f\}$;
10      **if** $(t = PREEMPTPATH)$ **then**
11        **ScheduleSuspend**($G$, $F_{active}$, $F_{suspend}$);
12      **end**
13     **end**
14     **if** $(t = NOPATH)$ **then**
15       $f.state \leftarrow$ SUSPENDED; $f.path \leftarrow p$;
16       $F_{suspend} \leftarrow F_{suspend} + \{f\}$ ;
17     **end**
18     **return**
     $\{< e.state, e.path >, \forall e \in F_{suspend} \cup F_{active}\}$;
19   **end**

---

When a flow $f$ in $F_{active}$ finishes, it is removed from $F_{active}$ and invokes the sub-algorithm ScheduleSuspend(.) to check

in turn whether the suspended flows in $F_{suspend}$ can become active (Lines 1-4). When a new flow $f$ arrives at the network, we use the sub-algorithm PathCalculation(.) to calculate the routing path and path type for the flow $f$ (Line 6). If the flow has an idle path or a preemptive path, it is added into $F_{active}$ (Lines 7-9) and scheduled immediately with the calculated routing path. Otherwise, flow $f$ is suspended (Lines 14-17). Note that if calculating a preemptive path for flow $f$, all the other flows in the suspended set require rescheduling (Lines 10-12) for two reasons. First, the preempted flows may have higher priorities than some active flows. Second, the preempted flows may leave available paths for some other suspended flows.

The sub-algorithms PathCalculation(.) and Schedule Suspend(.) are described in Algorithms 2 and 3, respectively. Algorithm 2 is responsible for calculating the routing path and path type for a target flow $f$. We first calculate multiple available routing paths for flow $f$ out of the network topology $G$ (Line 1), which is easy to get in regular, "richly-connected" data center network. For example, in a Fat-Tree network containing $k$ pods, there are $(k/2)^2$ available paths between two servers in different pods. In a level-$k$ BCube network, each pair of servers has $k + 1$ disjoint paths. Then we use the function SelectIdlePath(.) to choose an idle path for flow $f$ from the multiple candidates (Line 2). If there is no such path, we use the function SelectPreemptPath(.) to calculate a preemptive path for flow $f$ (Line 7). For the flows that share links with flow $f$ and have lower priorities than flow $f$ (Line 9), they are preempted and put to the suspended set (Lines 10-15). The details of calculating the preemptive path is shown in Algorithm 4, where the function FlowSetCrossPath($p$,$F_{active}$) is used to compute the set of active flows whose routing paths share links of path $p$. The function SelectMxPath(.) is to choose the path that traverses the minimum number of idle switches from multiple candidates.

---

**Algorithm 2.** PathCalculation($G$,$F_{active}$,$f$)

   **Input:**
   $G$: data center network topology;
   $F_{active}$: set of active flows;
   $f$: the target flow.
   **Output:**
   $\{< t, p >\}$: the type and ID of the path for flow $f$
1   $A \leftarrow$ GetAvailablePathSet($G$,$f$);
2   $B \leftarrow$ SelectIdlePath($A$,$F_{active}$);
3   **if** $(B \neq \phi)$ **then**
4     $b \leftarrow$ SelectMxPath($B$);
5     **return** $< IDLEPATH, b >$;
6   **else**
7     $c \leftarrow$ SelectPreemptPath($f$,$A$,$F_{active}$);
8     **if** $(c \neq NULL)$ **then**
9      $D \leftarrow$ PreemptedFlowSet($G$,$F_{active}$,$f$,$c$);
10     **for** each flow $d \in D$ **do**
11      $d.state \leftarrow$ SUSPENDED; $d.path \leftarrow -1$ ;
12      $F_{active} \leftarrow F_{active} - \{d\}$ ;
13      $F_{suspend} \leftarrow F_{suspend} + \{d\}$ ;
14     **end**
15     **return** $< PREEMPTPATH, c >$;
16     **else**
17      **return** $< NOPATH, -1 >$;
18     **end**
19   **end**

---

**Algorithm 3.** ScheduleSuspend($G$, $F_{active}$, $F_{suspend}$)

   **Input:**
   $G$: data center network topology;
   $F_{active}$: set of active flows;
   $F_{suspend}$: set of suspended flows.
   **Output:**
   $\{<e.state, e.path>, \ \forall e \in F_{suspend} \cup F_{active}\}$: scheduling state and path of each flow in $F_{suspend}$ and $F_{active}$
1   $x \leftarrow 1$;
2   **While** $x = 1$ **do**
3     $x \leftarrow 0$;
4     Sort $F_{suspend}$ by flow priority in a descending order;
5     $F'_{suspend} \leftarrow F_{suspend}$;
6     **for** each flow $e \in F'_{suspend}$ **do**
7      $< t, p > \leftarrow$ **PathCalculation**($G$,$F_{active}$,$e$);
8      **if** $(t = IDLEPATH \| t = PREEMPTPATH)$ **then**
9       $e.state \leftarrow$ ACTIVE; $e.path \leftarrow p$;
10      $F_{suspend} \leftarrow F_{suspend} - \{e\}$ ;
11      $F_{active} \leftarrow F_{active} + \{e\}$ ;
12      **if** $(t = PREEMPTPATH)$ **then**
13       $x \leftarrow 1$;
14       break;
15      **end**
16     **end**
17     **end**
18   **end**
19   **return**$\{< e.state, e.path >, \forall e \in F_{suspend} \cup F_{active}\}$;

---

**Algorithm 4.** SelectPreemptPath($f$,$A$,$F_{active}$)

   **Input:**
   $f$: target flow;
   $A$: set of available paths for $f$;
   $F_{active}$: set of active flows.
   **Output:**
   $t$: the preemptive path for $f$.
1   $t \leftarrow$ NULL;
2   $T \leftarrow \phi$;
3   **for** each path $p \in A$ **do**
1     $F' \leftarrow$ FlowSetCrossPath($p$,$F_{active}$) ;
5     $i \leftarrow 0$ ;
6     **for** each flow $f' \in F'$ **do**
7      **if** $(Priority(f) \leq Priority(f'))$ **then**
8       $i \leftarrow 1$ ;
9       break;
10      **end**
11     **end**
12     **if** $(i = 0)$ **then**
13      $T \leftarrow T + \{p\}$
14     **end**
15     $t \leftarrow$ SelectMxPath($T$);
16   **end**
17   **return** $t$;

---

When an active flow finishes transmission or is suspended due to preemption, the sub-algorithm Schedule-Suspend(.) will be invoked. It is used to check whether the existing suspended flows can be reactivated. In Algorithm 3, we first sort the suspended flows in a descending order by their priorities, and then calculate their routing paths and path types (Line 4). If a flow has an idle path or a preemptive path, it becomes active and

is scheduled immediately (Lines 8-11). Note that during the process some other active flows may be preempted and become suspended (Lines 12-14), which will update the set of suspended flows (refer to Lines 10-14 in Algorithm2). When the set of suspended flows changes, we need to immediately reschedule the flows in $F_{suspend}$ (Lines 12-15), for the same reason as that in Algorithm 1. The process ends when no more flows are added into the set of suspended flows (Lines 2-3).

We can run the EXR scheduling algorithm quickly (in polynomial time) at the SDN controller. The time it takes to perform EXR scheduling depends on different trigger conditions, i.e., the completion of a running flow or arrival of a new flow. For the former case, the scheduling time mainly comprises of invoking Algorithm 4 to schedule the suspended flows. For the latter one, the main step is calculating path for the new flow by invoking Algorithm 2, and schedule the suspended flows if necessary. The time complexity for Algorithm 2 is $O(|A| \cdot |F_{active}|)$, where $|A|$ is the number of available paths for the new flow (e.g., $\frac{1}{4}k^2$ in a $k$-ary Fat-Tree network). The time complexity for Algorithm 4 is $O(\Psi \cdot |A| \cdot |F_{active}| \cdot |F_{suspend}|)$, where $\Psi$ is the number of times needed to schedule suspended flows due to preemption of active ones during the process. $\Psi$ is bounded, since each round of scheduling ensures several flows with relative high priority are in $F_{active}$ and never get preempted later.

Note that EXR only adds the computation burden to the controller platform, instead of communication burden. Given that multi-core computation platform is developed and distributed controllers based on multiple computation entities are proposed, the potential scalability of EXR controller will not be a blocking problem for practical deployment.

## 4  SIMULATION

In this section, we evaluate EXR by simulations in typical data center topologies. We will show that compared with FSR, EXR has obvious advantages in saving network energy and reducing the average flow completion time if assigning higher scheduling priorities to smaller flows.

### 4.1  Simulation Setup

*Network topology.* We use three representative data center topologies, i.e., Fat-Tree, Blocking Fat-Tree and BCube. The capacity of each link is 1 Gbps. The oversubscription ratio of the original Fat-Tree network is 1:1, since it provides rearrangeble non-blocking communication service to end servers. As for the blocking Fat-Tree network, we vary its oversubscription ratio by cutting the number of core switches. In a blocking Fat-Tree with $k$:1 oversubscription ratio, the number of core switches is $1/k$ of that in the original Fat-Tree. Fig. 5 shows an example of a blocking Fat-Tree topology with a 2:1 oversubscription ratio. In all the Fat-Tree and blocking Fat-Tree topologies, we use switches with 24 ports, hence the total number of servers is 3,456. For BCube network, we use level-3 BCube with eight-port switches, in which the total number of servers is 4,096.

*Traffic flows.* Based on the real traces from production data centers, we assume the flow size follows an exponential distribution with the expected size of 64 MB, which is the typical size in distributed file system in data centers
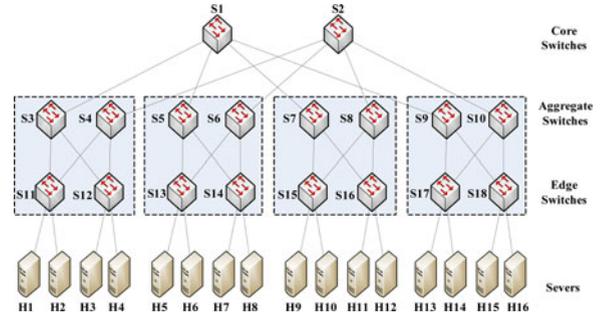


Fig. 5. A blocking Fat-Tree with oversubscription ratio of 2:1.

[16]. Note that the flow size can usually be obtained or estimated in advance for many data center applications, e.g., web search, key-value store and data processing, etc., which is also assumed in recent works [17], [18], [19]. The source and destination servers of each flow are randomly chosen. The arrival rate of the flows follows Poisson distribution. We simulate a total number of 10,000 flows.

*Power parameters.* We take the power parameters of a Cisco Nexus 2224TP switch in the simulation. The fixed part consumes 48 W while each port consumes 2 W [13]. The transition time between active state and sleeping state of switches/ports will account for the total network energy, and we study the impact in Section 4.4. The network energy is calculated using Eq. (1) in Section 2-A.

*Bandwidth sharing in FSR.* In FSR, each flow is scheduled to run immediately after arrival, and the flows use TCP congestion control algorithm to share the link bandwidths in a max-min fairness manner. ECMP routing is used to choose the next hop given multiple equal options.

*Flow priorities in EXR.* We take flow size as the parameter to assign the priorities. A smaller flow always has higher priority than a larger one.

*Flow completion time of EXR.* The flow completion time comprises the transmitting time and suspended time. Note that the time to calculate and configure flow paths is included in the suspended time, since the flow is suspended then.

### 4.2  Impact of Flow Arriving Rate

We first study the impact of flow arriving rate in the Poisson arrival. Figs. 6a and 6b show the network energy used to transmit all the flows against the flow arriving rate in the Fat-Tree and BCube networks, respectively. In Fig. 6a, we find that the Fat-Tree network energy decreases in both EXR and FSR as the flow arriving rate increases. It is because higher flow arriving rate provides more opportunities for concurrent flows to share the switches, and thus increases the switch utilization ratio and leads to more efficient network energy usage. EXR saves considerable network energy compared with FSR. For example, when the flow arriving rate is 500/second, EXR consumes 40 percent less network energy than FSR.

Similar results are shown in Fig. 6b. When the flow arriving rate varies from 300/second to 2,000/second, the BCube network energy decreases by more than 20 percent. The curve of BCube network is smoother than that of Fat-Tree network, because the switches in BCube network have much less ports than those in Fat-Tree network, hence the impact of switch multiplexing is lower than in Fat-Tree.
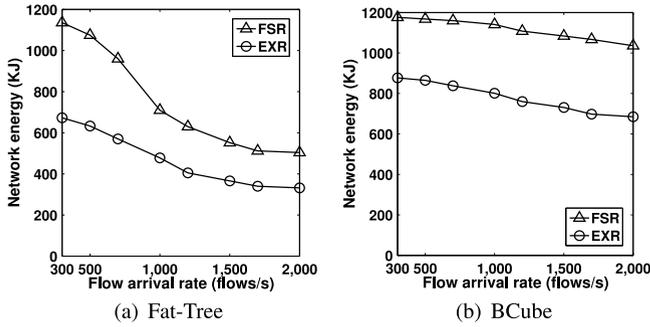
Fig. 6. Network energy against flow arriving rate in Fat-Tree and BCube networks.



Fig. 8. Network energy and average flow completion time in a blocking Fat-Tree against oversubscription ratio.

In BCube EXR also consumes much less power than FSR. When the flow arriving rate is 2,000/second, EXR saves about 36 percent network energy compared with FSR.

Figs. 7a and 7b show the average flow completion time against the flow arriving rate in Fat-Tree and BCube networks, respectively. We find that the average flow completion time ascends with the increase of flow arriving rates. It follows the intuition because when the flow arriving rate is higher, flows are more likely to compete the link bandwidths with others and as a result they will be transmitted by slower speeds. Compared with FSR, EXR effectively reduces the average flow completion time, in particular at high flow arriving rates. For example, when the flow arriving rate is 2,000/second, the average flow completion time in EXR is about 60 percent of that in FSR, in both Fat-Tree and BCube networks.

### 4.3   Impact of Network Oversubscription Ratio

The non-blocking Fat-Tree topology enjoys high bisection bandwidth and 1:1 oversubscription ratio. However, most of the current data centers still employ blocking multi-rooted tree topologies [20]. Therefore, next we study the energy conservation and flow completion time of EXR in blocking Fat-Tree with different oversubscription ratios. We proportionally eliminate the core switches from the non-blocking Fat-Tree to build the blocking Fat-Tree, and show the simulation results with oversubscription ratios of 1:1, 2:1, 4:1, 8:1 and 16:1, respectively. In this group of simulations, we set the average flow arriving rate as 1,000/second.

Fig. 8a demonstrates that EXR effectively reduces the network energy consumption in blocking Fat-Tree topologies, and the benefit is more obvious with higher oversubscription ratio. The network energy in either EXR or
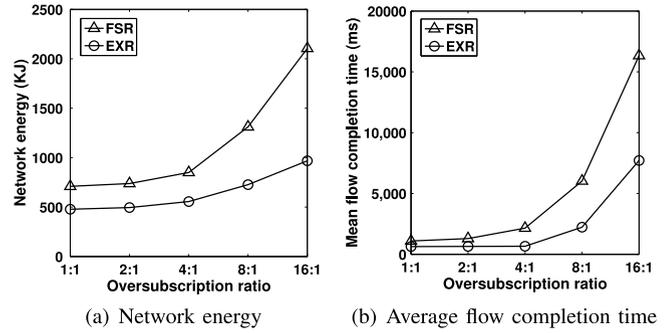
FSR rises with the increase of the oversubscription ratio. It is because when the network is highly blocked, the traffic loads in active switches are quite unbalanced and many active switches are under-utilized. EXR performs even better than FSR in higher-oversubscribed network, because the active links in EXR are always fully utilized. When the oversubscription ratio is 16:1, EXR can save more than 50 percent of the overall network energy compared with FSR.

Fig. 8b shows the average completion time of the flows, which increases as the oversubscription ratio gets higher. Higher oversubscription indicates more competition among flows. When the oversubscription ratio increases to 16:1, the average flow completion time in FSR is almost twice that in EXR.

### 4.4   Impact of Power State Transition Time

In the simulations above, we do not take the power state transition time of switches and ports into account, and ideally assume that they can go to sleep soon upon idleness. However, in practice the transition time between active mode and sleep mode cannot be simply ignored. It usually takes tens of microseconds to complete a sleep/wake transition for an Ethernet port and several seconds for a data center switch [14], [21], [22].

Next we study the impact of the power state transition time in EXR. We set the average flow arriving rate as 1,000/second, and assume that switches and ports can go to sleep only if their sleeping durations are longer than the transition time required.

Figs. 9a and 9b show the network energy with EXR and FSR against different power state transition times of switches in Fat-Tree and BCube, respectively. We fix the transition time of each port as 10 milliseconds. The results
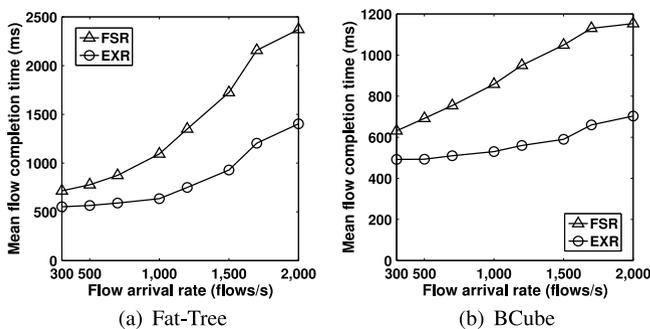


Fig. 7. Average flow completion time against flow arriving rate in Fat-Tree and BCube networks.
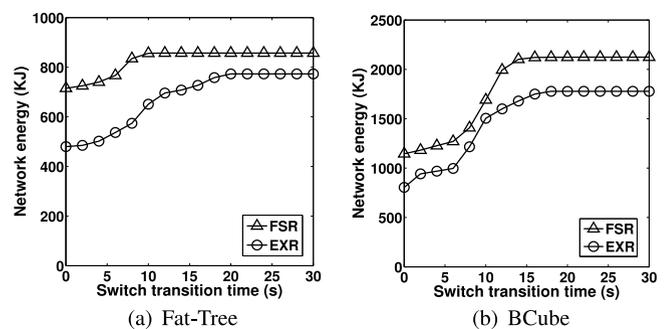


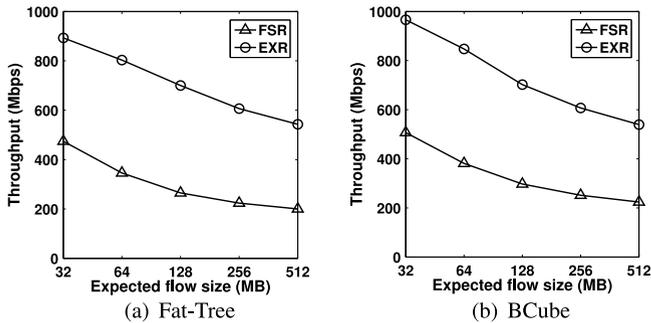Fig. 9. Network energy against the power state transition time of switches in Fat-Tree and BCube networks.

Fig. 10. Average flow throughput against the expected flow size in Fat-Tree and BCube networks.



Fig. 11. Network energy against the expected flow size in Fat-Tree and BCube networks.

in the two figures demonstrate that EXR enjoys much less network energy than FSR, under whatever power state transition times of switches. For example, when the switch's power transition time is 2 seconds, EXR saves more than 30 percent and more than 20 percent network energy compared with FSR, in Fat-Tree and the BCube networks respectively. The network energy also increases when the power transition time is longer, because higher transition time will prevent switches with short idle durations from sleeping. Besides, we observe that when the switch's power transition time is longer than a threshold, the network energy consumed becomes constant. It is because in these cases no switches can go to sleep during the flow transmission process, while the energy saving of EXR comes from that the flows can finish transmission earlier than FSR.

### 4.5 Impact of Network Traffic Load

To achieve network energy efficiency, the energy consumption at traffic valley time should be reduced by means of setting idle switches and links to sleep mode, which is shown to be one property of EXR previously. However, during traffic peak time, the focus should be on the maximum capacity offered by the network to support the heavy traffic, since there is not too much room for energy saving and the network naturally consumes more energy than that at traffic valley time.

Hence, we study impact of network traffic load on EXR in terms of throughput and energy consumption. Assume the average flow arriving rate is 1,000/second, we gradually increase flow size and show the simulation results with expected flow size of 32, 64, 128, 256 and 512 MB, respectively.

Figs. 10a and 10b show the average throughput perceived by each flow with EXR and FSR against the expected flow size in Fat-Tree and BCube networks. In Fig. 10a, throughput in both EXR and FSR decreases when expected flow size increases. The reason is as follow. For FSR, more large flows coexist in the network and share the bottleneck link bandwidth, hence, the bandwidth share of each flow is smaller based on the TCP congestion control mechanism. For EXR, larger flow size leads to longer queuing delay, since suspended flows have to wait longer before becoming active and transfer data at full rate. Besides, the average throughput of EXR is much higher than FSR in all cases, due to the elimination of congested links. Fig. 10b shows
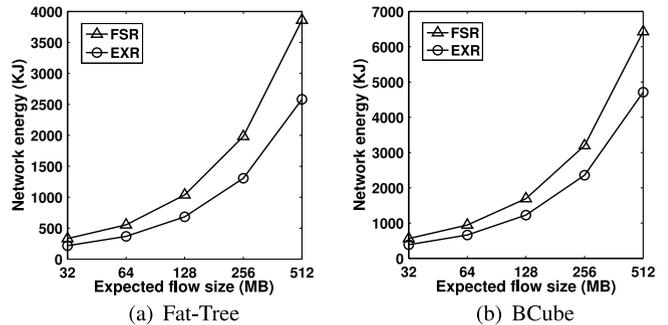
similar results in BCube network, with throughput being slightly higher as a result of structural difference of Fat-Tree and BCube topology.

Figs. 11a and 11b demonstrate that EXR can effectively reduce the network energy though the room for energy saving is limited at traffic peak time. When average flow size is larger, the switches and links have to work longer to support the heavy traffic, thus consuming more energy. EXR saves energy by finishing flows earlier and turn switches into sleep mode upon idleness. For instance, when average flow size is 512 MB, mean flow completion time of EXR reduces 35.02 and 26.56 percent relative to FSR in Fat-Tree and BCube networks.

## 5 IMPLEMENTATION AND EXPERIMENTS

We implement both EXR and FSR by modifying SDN/OpenFlow protocol. For both the scheduling approaches, when a new flow is generated by a virtual machine and arrives at the first-hop switch (virtual switch in the hypervisor), it is directed to the controller. For FSR, the controller directly configures the forwarding entries in the switches along the flow's routing path and replies to the virtual switch to permit the flow. For EXR, the controller should calculate the routing path and scheduling state for the new flow. If the flow is in active state, the controller also immediately configures the forwarding tables in corresponding switches and notifies the virtual switch to permit the flow; otherwise, the controller notifies the virtual switch to block the flow. In EXR, the virtual switch also monitors the complete of flow transmission and reports to the controller, which will also trigger the EXR flow scheduling. Given any change of the flows' states, the controller directly tells the virtual switch to permit a flow or block a flow.

We conduct experiments in a partial Fat-Tree test bed, which contains two pods and eight servers. The topology is the same with Fig. 2. All the servers are Desktops with 2.80 GHz Intel Duo E7400 processors and 2 GB of RAM, and the network cards in the servers are Intel 82574L Gigabit Ethernet cards. H3C S5500-24P-SI series switches are employed to connect the servers in the testbed. We use iperf [23] to generate 16 flows, with average size of 500 MB and arrival interval of 1 second. The source and destination servers of the flows are chosen randomly. We run EXR and FSR respectively in the controller to schedule these flows, and show the experiment results in Fig. 12.
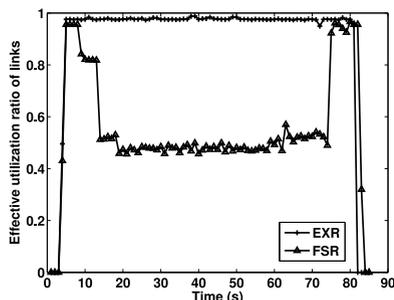
Fig. 12. Effective utilization ratio of links in a Fat-Tree testbed.

We measure the metric of effective link utilization, which indicates the average utilization ratio of all the active links. As aforementioned, a higher effective utilization ratio of links indicates more efficient energy usage of switches/ ports. In Fig. 12, we find that the effective link utilization ratio under EXR almost reaches 100 percent at most time. It is because when using the EXR scheme, there is no competi- tion of link bandwidth among flows. Hence, every flow can fully utilize the link bandwidth of its routing path. In con- trast, when using FSR, the average effective link utilization ratio only hovers at around 50 percent at most time, due to flow competition in bottleneck links and under-utilization in other links.

The experiment results demonstrate that EXR can make flows sufficiently utilize the bandwidths of active links, which helps improve the energy efficiency of data center networks and save the overall network energy given the same traffic load.

# 6  DISCUSSION

*Energy efficiency.* EXR achieves energy efficiency through saving network energy at traffic valley time and exploiting the highest network capacity when the traffic load is heavy. When the network is under loaded, it's unnecessary for all the switches to stay active to support the flows. Hence, switches can go to sleep for energy conservation if their idle durations exceed a predefined threshold (i.e., power state transition time). Since the room for energy conservation is quite limited at heavy traffic load time, routing schemes should focus on improving the throughput of flows. EXR explores the limited room by scheduling flows to fully (and exclusively) utilize link bandwidth, and results in high aver- age flow throughput as well as low flow completion time. On the contrary, FSR schedules flows in a max-min fairness manner, which prolongs flow transmission process due to low per-flow bandwidth share. It's worth noting that EXR choose the paths with minimum number of idle switches to save energy, this wouldn't cause congestion, as flows exclu- sively occupy the links.

*Flow priority.* EXR leaves the flexibility to operators to assign flows with different scheduling priorities [18]. When the Earliest Deadline First (EDF) strategy is applied, number of late flows that miss their deadline is minimized. If we adopt Shortest Job First (SJF) strategy instead, the average flow completion time can be reduced and the throughput is higher. However, none far surpasses the other in practice with different kinds of flows. For delay sensitive flows, EDF is apparently more suitable. For flows that take throughput

as a critical performance parameter (i.e., video streaming), SJF lowers the average flow completion time and has higher throughput, hence is more appropriate. Thanks to the emerging SDN technology, flow priority is only used at the SDN controller and no switches or servers need to know about it, not to mention setting a priority field (i.e., ToS field in IP headers) in each flow packet. Besides, in practical data centers, there are mainly two types of flows [24], i.e., small/ mouse flows and big/elephant flows, with different require- ments. Mouse flows require fast response and low latency, while elephant flows require high throughput. Hence, mouse flows are supposed to be given higher priorities. In order to alleviate flow starvation problem of elephant flows, operators can flexibly assign flows with hybrid priorities by taking the suspended time into consideration. The longer an elephant flow has been suspended, the higher priority it should obtain.

*Short lived flows.* Intuitively, one may think that EXR will be less attractive for small flows that last only a short time, i.e., the idle time gaps of switches and links is rela- tively smaller than the power state transition time in this situation, which prevents switches and links with short idle durations from sleeping. However, in Section 4.4, we show that EXR still saves energy in this case. Though fewer switches and links go to sleep for energy conserva- tion during flow transmission process, EXR enjoys higher link utilization and higher throughput to ensure that flows finish earlier than FSR. Therefore, the switches and links can go to sleep immediately for energy saving when all flows are finished, while they have to stay active to support remaining flows if FSR is applied.

*Network-limited flows and application-limited flows.* In the design of EXR, we focus on network-limited flows, in which the flows generated by applications can saturate the band- width capacity of links fast enough. Actually, the data cen- ter networks are typically oversubscribed, with an oversubscription ratio of 2.5:1 to 8:1 [5], to lower the cost of network architecture. Hence, the network is often the bottle- neck that hinders the flow transmission process. However, one may naturally wonder how EXR will perform in the opposite scenario, i.e., the application-limited flows, in which the bandwidth capacity of links are capable of accom- modating flows generated by applications at full speed. As mentioned in Section 3.2, EXR can be easily extended to application-limited flows, by putting multiple flows in a link if the available bandwidth allows. More specifically, when choosing the path for a new flow, we relax the con- straint of exclusively occupying the whole link, whose bandwidth capacity the flow can't fully utilize, due to its application-limited nature. The residual link capacity can be utilized by the suspended flows, i.e., multiple flows share the same link provided that they don't exceed the band- width capacity. In order to present EXR in a clean way, this paper focuses on network-limited flows.

*Flow statistical multiplexing.* FSR improves the link utiliza- tion by flow statistical multiplexing, i.e., multiple flows share the same link simultaneously. However, as demonstrated in Section 2.2, FSR results in low utilization as a whole, due to inevitable bandwidth competition among flows. Flows fairly share the bandwidth of bottleneck links by TCP congestion control, and only improves the utilization of the bottleneck

links ($\leq 10$ percent, see Section 2.2). However, a great number of non-bottleneck links are under-utilized. EXR enjoys higher link utilization since flows fully/exclusively occupy the active links, and inactive links are put to sleep mode for energy saving. Besides, EXR still has flow statistical multiplexing, in the time dimension. When multiple flows have to share the same link, they are scheduled one after another without time overlap.

## 7 RELATED WORKS

### 7.1 Greening Networks

#### 7.1.1 Device-Level Techniques

The device-level energy conservation schemes aim to improve the energy efficiency of individual network devices with advanced hardware techniques. Nedevschi et al. [22] claimed that the energy consumption of network devices could be effectively improved by applying the sleep-on-idle and the rate-adaptation technique. Gupta and Singh [25] studied the time to put idle and under-utilized Ethernet interfaces to the low-power mode, based on the number of arriving packets in a given time period, and analyzed the tradeoff between energy conservation and packet loss and delay. Similarly, Gunaratne et al. [26] designed the policies of link rate adaption by monitoring the link utilization and the queue length of output buffers to effectively save the Ethernet link energy. The authors in [12] designed shadow ports to buffer ingress packets, and used the schemes of time window prediction and power save mode to reduce the energy consumption of switch ports.

#### 7.1.2 Network-Wide Techniques

The network-wide energy conservation schemes study how to reduce the gross network energy consumption from the perspectives of topology design and routing. Chabarek et al. [27] formulated the problem of minimizing network energy consumption as a multiple commodity network flow problem, and investigated the energy-efficient network and protocol designs with mixed-integer program techniques. Vasić et al. [28] proposed a two-step flow scheduling scheme to balance the optimal energy conservation and the computational complexity. In the scheme, a group of energy-critical paths were pre-computed and a scalable traffic engineering mechanism was then used to schedule network flows in an online manner. Abts et al. [4] proposed an energy-efficient flattened butterfly topology and reconfigured the transfer rates of links periodically to make data center networks more energy proportional. Heller et al. [10] designed a network energy optimizer, namely, ElasticTree, which studied how to choose a subset of links to transmit traffic flows under the restrictions of network performance and reliability. Furthermore, Shang et al. [11] designed a throughput-guaranteed power-aware routing algorithm, which used a pruning-based method to iteratively eliminate switches from the original topology while meeting the network throughput requirement.

### 7.2 Flow Scheduling in Data Center Networks

Many flow scheduling schemes have been proposed in data center networks to achieve different optimization objectives, such as maximizing network utilization, minimizing flow completion time and meeting the deadline constraints of network flows.

Al-Fares et al. [29] proposed a dynamic flow scheduling scheme, i.e., Hedera, for multi-rooted hierarchical tree topologies used in data center networks. It can efficiently utilize link resources without significant compromise on the control and computation overheads. The experiment results demonstrated that Hedera achieves higher bisection bandwidth than ECMP routing. Wilson et al. [17] presented a deadline-aware control protocol, named $D^3$, which controlled the transmission rate of network flows according to their deadline requirements. $D^3$ can effectively improve the latency of mice flows and burst tolerance, and increase the transmission capacity of data center networks. Hong et al. [18] proposed a preemptive distributed quick flow scheduling protocol, i.e., PDQ, which aimed to minimize the average completion time of network flows as well as to meet their deadlines. The evaluation results showed that PDQ outperformed TCP and $D^3$ [17] in terms of the average flow completion time and the number of flows meeting deadlines. Zats et al. [30] proposed DeTail, which designed a cross-layer network stack aiming to improve the tail of completion time for delay-sensitive flows.

## 8 CONCLUSION

In this paper, we explored a new flow scheduling method to save the energy consumed by data center networks, called EXR. The essence of EXR is to schedule traffic flows in the time dimension and let each flow exclusively occupy the link bandwidths on its routing paths. The advantage of EXR in saving more network energy comes from that it removes bottleneck active links in data center networks and improves the utilization ratio of active switches/links. Besides saving network energy, EXR can also help reduce the expected flow completion time if assigning higher scheduling priorities to smaller flows, requires neither time synchronization among servers nor bandwidth reservation for flows, etc. Simulation and experiment results show that compared with FSR, EXR can effectively save network energy and shorten the average flow completion time.

## REFERENCES

[1] J. Koomey, "Growth in data center electricity use 2005 to 2010," whiter paper, in *Analytics Press*, Aug. 2011.
[2] P. X. Gao, A. R. Curtis, B. Wong, and S. Keshav, "It's not easy being green," *ACM SIGCOMM 2012*, Helsinki, Finland, Aug. 2012.
[3] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, pp. 68–73, Jan. 2009.

[4] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *Proc. 37th Annu. Int. Symp. Comput. Archit.*, 2010, pp. 338–347.

[5] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2008, pp. 63–74.

[6] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: A high performance, server-centric network architecture for modular data centers," in *Proc. ACM SIGCOMM Conf. Data Commun.*, 2009, pp. 63–74.

[7] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, and S. Lu, "FiConn: Using backup port for server interconnection in data centers," in *Proc. IEEE INFOCOM*, 2009, pp. 2276–2285.

[8] D. Li, M. Xu, H. Zhao, and X. Fu, "Building mega data center from heterogeneous containers," in *Proc. IEEE 19th IEEE Int. Conf. Netw. Protocols*, 2011, pp. 256–265.

[9] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, "A power benchmarking framework for network devices," in *Proc. 8th Int. IFIP-TC 6 Netw. Conf.*, 2009, pp. 795–808.

[10] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "ElasticTree: Saving energy in data center networks," in *Proc. 7th USENIX Conf. Netw. Syst. Des. Implementation*, Apr 2010, p. 17.

[11] Y. Shang, D. Li, and M. Xu, "Energy-aware routing in data center network," in *Proc. ACM SIGCOMM Workshop Green Netw.*, 2010, pp. 1–8.

[12] G. Ananthanarayanan and R. H. Katz, "Greening the switch," in *Proc. Conf. Power Aware Comput. Syst.*, 2008, p. 7.

[13] Cisco Nexus 2200 series data sheet. [Online]. Available: http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps10110/data _sheet_c78-507093.html, 2013.

[14] IEEE P802.3az energy efficient ethernet task force. [Online]. Available: www.ieee802.org/3/az/index.html, 2014.

[15] C. Hopps, "Analysis of an equal-cost multi-path algorithm," RFC 2992, IETF, 2000.

[16] S. Ghemawat, H. Gobioff, and S. Leung, "The Google file system," in *Proc. 19th ACM Symp. Oper. Syst. Principles*, 2003, pp. 29–43.

[17] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," in *Proc. ACM SIGCOMM Conf.*, 2011, pp. 50–61.

[18] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," in *Proc. ACM SIGCOMM*, 2012, pp. 127–138.

[19] B. Vamanan, J. Hasan, and T. N. Vijaykumar, "Deadline-aware datacenter tcp (d2tcp)," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Archit. Protocols Computer Commun.*, 2012, pp. 127–138.

[20] Cisco Data Center Infrastructure 2.5 Design Guide. [Online]. Available: http://www.cisco.com/application/pdf/en/us/guest/netsol/ns107/c649/ccmi gration_09186a008073377d.pdf, 2014.

[21] IEEE 802.3az energy efficient ethernet: build greener networks. [Online]. Available: www.cisco.com/en/US/prod/collateral/switches/ps5718/ps4324/white_paper_c11-676336.pdf, 2014.

[22] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via sleeping and rate-adaptation," in *Proc. 5th USENIX Symp. Netw. Syst. Des. Implementation*, 2008, pp. 323–336.

[23] Iperf. [Online]. Available: http://iperf.sourceforge.net/, 2014.

[24] Y. Cao, M. Xu, X. Fu, and E. Dong, "Explicit multipath congestion control for data center networks," in *Proc. 9th ACM Conf. Emerging Netw. Exp. Technol.*, 2013, pp. 73–84.

[25] M. Gupta and S. Singh, "Using low-power modes for energy conservation in ethernet LANs," in *Proc. IEEE 26th Int. Conf. Comput. Commun.*, 2007, pp. 2451–2455.

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

[27] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power awareness in network design and routing," in *Proc. IEEE 27th Conf. Comput. Commun.*, 2008, pp. 1130–1138.

[28] N. Vasić, P. Bhurat, D. Novaković, M. Canini, S. Shekhar, and D. Kostić, "Identifying and using energy-critical paths," in *Proc. ACM 7th Conf. Emerging Netw. Exp. Technol.*, 2011, pp. 205–212.

[29] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. 7th USENIX Conf. Netw. Syst. Des. Implementation*, 2010, p. 19.

[30] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "Detail: Reducing the flow completion time tail in datacenter networks," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Archit. Protocols Comput. Commun.*, 2012, pp. 139–150.

**Dr. Dan Li** received the PhD degree in computer science from Tsinghua University in 2007. He is an associate professor in the Computer Science Department of Tsinghua University, Bejing, China. His research interest includes Future Internet architecture and data center networking.

**Mr. Yunfei Shang** is working towards the PhD degree in the Computer Science Department of Tsinghua University, China. His main research interest includes data center networks and green networking.

**Mr. Wu** received the BS degree in 2012 from the Department of Computer Science and Technology, Beijing Normal University. Currently, he is a graduate student in Beijing Normal University and a visiting graduate student in Tsinghua University. His research interests include data center network and network virtualization.

**Mr. Congjie Chen** is a master's student in the Computer Science Department of Tsinghua University, China. His main research interest includes data center networks and network virtualization.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.