Profit-Aware Distributed Online Scheduling for Data-Oriented Tasks in Cloud Datacenters

Wei Lu, Ping Lu, Quanying Sun, Shui Yu, *Senior Member, IEEE*, Zuqing Zhu, *Senior Member, IEEE*

***Abstract*—As there is an increasing trend to deploy geographically distributed (geo-distributed) cloud datacenters (DCs), the scheduling of data-oriented tasks in such cloud DC systems becomes an appealing research topic. Specifically, it is challenging to achieve the distributed online scheduling that can handle the tasks' acceptance, data-transfers and processing jointly and efficiently. In this paper, by considering the store-and-forward and anycast schemes, we formulate an optimization problem to maximize the time-average profit from serving data-oriented tasks in a cloud DC system and then leverage the Lyapunov optimization techniques to propose an efficient scheduling algorithm, *i.e.*, *GlobalAny*. We also extend the proposed algorithm by designing a data-transfer acceleration scheme to reduce the data-transfer latency. Extensive simulations verify that our algorithms can maximize the time-average profit in a distributed online manner. The results also indicate that *GlobalAny* and *GlobalAny_Ext* (*i.e.*, *GlobalAny* with data-transfer acceleration) outperform several existing algorithms in terms of both time-average profit and computation time.**

***Index Terms*—Datacenter networks, Lyapunov optimization, Distributed online scheduling, Data-transfer acceleration.**

I. Introduction

NOWADAYS, there is an increasing trend to deploy geographically distributed (geo-distributed) cloud datacenters (DCs) for achieving enhanced user experience and service availability [1, 2]. In such cloud DCs, emerging applications usually involve numerous data-oriented tasks [3–7]. These tasks need to transfer certain amount of data to remote DC(s) for further processing. For instance, e-Science and backup applications may need to transfer data among multiple geo-distributed DCs [8, 9]. As the data-oriented tasks can cause high-capacity data-transfer among DCs, they would impact the service capacity and resource management in cloud DC systems significantly. Therefore, we need to schedule them adaptively to ensure efficient DC operations.

The scheduling of data-oriented tasks bears some interesting features. First of all, the data-transfers can utilize the anycast routing scheme, *i.e.*, data can be forwarded to and processed by any DC within a destination DC set. For example, the data backup tasks can use the mutual backup model [1] and choose any DC within a backup group as its backup site. Secondly, the data-transfers usually can tolerate certain setup delay [10–13]. Hence, they can leverage the store-and-forward scheme [10]. Specifically, the intermediate DCs along the routing path can store the data when the subsequent network link is congested, and send it out to the destination DC later. Note that, these features make the scheduling of data-oriented tasks more flexible, and hence we need to incorporate more sophisticated scheduling algorithms. Moreover, the tasks are usually highly-dynamic, and their arrival pattern can hardly be predicted. Therefore, we need to find the optimal scheme to schedule them in an online manner without much pre-knowledge on the arrival pattern, which is obviously challenging.

In this work, by considering the anycast and store-and-forward schemes, we investigate how to allocate network resources and IT resources to support the data-oriented tasks dynamically and cost-effectively, *i.e.*, maximizing the time-average profit. Here, the profit is the margin between the revenue from task serving and the cost due to resource consumption. We formulate the optimization problem and design a distributed online scheduling algorithm, *i.e.*, *GlobalAny*, by leveraging the Lyapunov optimization techniques [14]. With *GlobalAny*, the destination DC of a task can be adjusted adaptively during the data-transfer, rather than being determined in its source DC directly. Our proposed algorithm maximizes the time-average profit in a distributed online manner and obtains the profit that is arbitrarily close to the optimal value. We then extend it by designing a data-transfer acceleration scenario to unify the benefits of direct-transfer and store-and-forward schemes. Finally, extensive simulations are performed to evaluate the performance of the algorithms and the results indicate that they can outperform several existing algorithms in terms of both time-average profit and computation time.

The rest of the paper is organized as follows. Section II reviews the related work, and we formulate the problem in Section III. The distributed scheduling algorithm is proposed in Sections IV. Section V describes the data-transfer acceleration scenario and the performance evaluation is discussed in Section VI. Finally, Section VII summarizes the paper.

II. Related Work

Previously, people have studied the task scheduling across cloud DCs with the objectives of load balancing [15] and budget minimizing [16]. Meanwhile, task scheduling was also considered for the Big Data applications [17–19]. However, these investigations did not address the data-transfer process in inter-DC networks, and might lead to long processing latency.

In order to handle inter-DC traffic well, researchers have considered the store-and-forward scheme for traffic scheduling [10, 20]. The authors of [10] studied the store-and-forward and time-expanded networking schemes for minimizing data-transfer time. The work in [20] tried to maintain the service

W. Lu, P. Lu, Q. Sun and Z. Zhu are with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, P. R. China. E-mail: zqzhu@ieee.org.

S. Yu is with the School of Information Technology, Deakin University, VIC 3125, Australia. Email: syu@deakin.edu.au.

Manuscript received January 10, 2018.

fairness for data-transfers with the store-and-forward scheme. Note that, these studies optimized the scheduling of data-transfers based on the pre-knowledge of their arrival patterns, and thus could only get the optimal solution for certain time period. Moreover, the computational complexity of their approaches increased significantly with the length of the period.

From the perspectives of network framework and protocol, the studies in [21–24] and [25–28] have considered how to realize virtual network embedding (VNE) and network function virtualization (NFV) in inter-DC networks, respectively, and Kettimuthu *et al.* [29] have tried to optimize the data-transfers in cloud DC systems. With the software-defined networking (SDN) based scheduling schemes, Google and Microsoft have developed systems such as B4 [12] and SWAN [30], respectively, to manage the inter-DC traffic in their DCs. Wu *et al.* investigated the data-transfers across SDN-controlled geo-distributed DCs in [31]. Our work is different from these studies, as we do not require centralized network control and management (NC&M) and the proposed algorithms can be implemented in a truly distributed way.

It is known that Lyapunov optimization techniques can be used to design online scheduling algorithms that can arbitrarily close to the time-average optimum without any pre-knowledge on the requests' arrival patterns [14]. Hence, they have been used to solve the scheduling problems in cloud computing [32], DC power management [33], *etc*. In [34], Liu *et al.* leveraged the Lyapunov optimization techniques to address the task scheduling and server/virtual machine (VM) management within a single DC. However, they did not consider how to determine the routing path and bandwidth allocation for the data-transfer of each task, which is essential for scheduling the data-transfers in inter-DC networks. Hence, their algorithms cannot be used to solve the problem studied in this work. Previously, with the Lyapunov optimization techniques, people have proposed the back-pressure algorithm to manage the data-transfers in multi-hop wireless networks [35]. However, it is known that this scheme can cause long and unnecessary data-transfer latency, especially when the traffic load is relatively low. Even though the data-oriented tasks can be delay-tolerant, obsessively long latency can still impact user experience. Hence, the data-transfer latency of the back-pressure algorithm has been addressed in [36, 37]. In [36], the authors tried to perform routing path selection based on the information of the shortest paths, while the technique based on shadow queues was proposed in [37], to shorten the data-transfer latency. However, these studies were still based on the hop-by-hop store-and-forward scheme, which cannot reduce the data-transfer latency to the maximum extent.

III. Problem Formulation

A. Network Model

We consider a geo-distributed cloud DC system as shown in Fig. 1. To facilitate distributed online task scheduling, the system operates based on discrete time-slot (TS) with a duration of Δt. Note that in a real cloud DC system, Δt can range from a few seconds to several minutes [34], depending on the traffic dynamics. Specifically, it should be

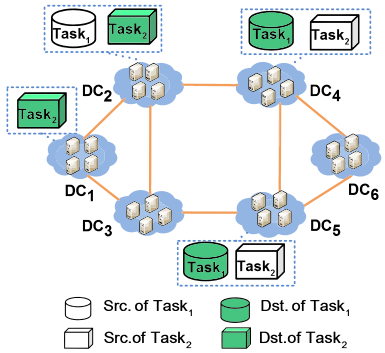

Fig. 1. Geo-distributed cloud DC system that carries data-oriented tasks.

sufficiently long for the system to implement the updates from the scheduling algorithm, *w.o.l.g.*, we normalize TS in the rest of the paper. Hence, we have $\Delta t = 1$, and the system time t is $t \in \{1, 2, \cdots\}$. There are multiple DCs in the system, and we denote the network as $\mathcal{G}(\mathcal{D}, \mathcal{E})$, where $\mathcal{D}$ is the DC set ($\mathcal{D} = \{1, \ldots, |\mathcal{D}|\}$) and $\mathcal{E}$ represents the set of inter-DC links. Generally, cloud DC provides numerous computing, storing or I/O resources in the form of VM for task processing. In this paper, we use IT resources to represent the resources that would be used to process the data-oriented tasks and assume the available IT resources in *DC* i is C_i. The available bandwidth on link $(i, v) \in \mathcal{E}$ is $B_{i,v}$, where i and v represent two adjacent DCs in the network ($i, v \in \mathcal{D}$).

In order to mimic the real cases, we assume that the data-oriented tasks in the cloud DC system can be classified into $|\mathcal{K}|$ categories and denote the set of categories as $\mathcal{K}$. Each category has its own source and destination DC sets and data-size. Specifically, for the k-th category, the tasks are generated in a DC in source DC set $\mathcal{S}_k \subset \mathcal{D}$ to deliver a block of δ_k data to any DC in destination DC set $\mathcal{D}_k \subset \mathcal{D}$ for further processing. For instance, for the first category in Fig. 1, the tasks originating from *DC* 2 need to deliver certain data to *DC* 4 or 5. Apparently, as we consider inter-DC traffic, the source and destination DC sets should not overlap, *i.e.*, $\mathcal{S}_k \cap \mathcal{D}_k = \emptyset, \forall k \in \mathcal{K}$. A weight coefficient w_k is assigned to the k-th category to facilitate differentiated services. We use $\mathcal{K}_i$ to represent the set of categories that can be processed in *DC* i. Table I lists the key notations used in the paper.

We use A_k^{max} to denote the maximum number of k-th category tasks generated per TS. Then, the arrival rate for the k-th category tasks in *DC* i at time t (*i.e.*, $A_i^k(t)$) satisfies

$$0 \le A_i^k(t) \le A_k^{max} \cdot \mathcal{I}_{\{i \in \mathcal{S}_k\}} \quad \forall i, k, \tag{1}$$

where $\mathcal{I}_{\{i \in \mathcal{S}_k\}}$ is a boolean flag to indicate whether *DC* i is included in source DC set $\mathcal{S}_k$, as

$$\mathcal{I}_{\{i \in \mathcal{S}_k\}} = \begin{cases} 1, & \forall i \in \mathcal{S}_k, \\ 0, & \forall i \notin \mathcal{S}_k. \end{cases}$$

B. Data-Oriented Tasks Scheduling in DCs

1) Task Acceptance: With the anycast scheme, the destination DC of a task can be adjusted by any of the intermediate DCs along its routing path, before it finally reaches a feasible

TABLE I
KEY NOTATIONS

$A_i^k(t)$	number of k-th category tasks generated in *DC* i at time t
$a_i^k(t)$	number of k-th category tasks accepted in *DC* i at time t
$Q_i^k(t)$	queue for k-th category tasks in *DC* i at time t
$X_i^k(t)$	virtual queue for k-th category tasks in *DC* i at time t
$x_i^k(t)$	number of tasks accepted for $X_i^k(t)$ at time t
$b_{i,v}^k(t)$	amount of k-th category data that is sent over link (i,v) at time t
$c_i^k(t)$	amount of IT resources allocated to k-th category tasks in *DC* i at time t

destination DC. As the destination DCs are selected in a global manner, we name this scheduling scheme as *GlobalAny*. Specifically, each DC uses the scenario illustrated in Fig. 2 to handle the data-oriented tasks. In each DC, we determine whether a locally-generated task should be accepted or not based on the status of a few queues, each of which stores the tasks to a remote DC. We arrange the queues in each DC based on the tasks' categories, *i.e.*, $Q_i^k(t)$ denotes the queue that stores all the tasks that belong to the k-th category in *DC* i and it is initialized as $Q_i^k(0)=0$. With all the queues in the cloud DC system (*i.e.*, $\{Q_i^k(t), i\in\mathcal{D}, k\in\mathcal{K}\}$), we can obtain the queue matrix $\mathbf{Q}(t)$ as

$$\mathbf{Q}(t)=\begin{bmatrix} Q_1^1(t) & Q_1^2(t) & \cdots & Q_1^{|\mathcal{K}|}(t) \\ Q_2^1(t) & Q_2^2(t) & \cdots & Q_2^{|\mathcal{K}|}(t) \\ \vdots & \vdots & \ddots & \vdots \\ Q_{|\mathcal{D}|}^1(t) & Q_{|\mathcal{D}|}^2(t) & \cdots & Q_{|\mathcal{D}|}^{|\mathcal{K}|}(t) \end{bmatrix}. \tag{2}$$

In *DC* i, the number of accepted k-th category tasks at time t is denoted as $a_i^k(t)$, and we have

$$0\le a_i^k(t)\le A_i^k(t),\quad \forall i,k. \tag{3}$$

The time-average expectation of $a_i^k(t)$ is

$$a_i^k=\lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{a_i^k(\tau)\},\quad \forall i,k.$$

2) Data-Transfer with Store-and-Forward Scheme: The data-transfer in the cloud DC system can be realized with the store-and-forward scheme [10], *i.e.*, by utilizing the storage space in the intermediate DCs along the routing path to relay the data hop-by-hop. It is known that this scheme can provide higher data-transfer throughput than the one that delivers the data end-to-end directly [10].

For data-transfer, a decision variable $b_{i,v}^k(t)$ is defined as the amount of k-th category data that is sent over link (i,v). Here, *DCs* i and v are adjacent in $\mathcal{G}(\mathcal{D},\mathcal{E})$. Apparently, the total transferred data should not exceed the available bandwidth on each link, *i.e.*,

$$\sum_k b_{i,v}^k(t)\le B_{i,v},\quad \forall i,\ \forall(i,v)\in\mathcal{E}. \tag{4}$$

The time-average expectation of $b_{i,v}^k(t)$ is denoted as $b_{i,v}^k$. And the total used bandwidth $b_{i,v}(t)$ can be obtained by summarizing the bandwidth used by all the task categories, and its time-average value $b_{i,v}$ should satisfy

$$b_{i,v}=\sum_k b_{i,v}^k,\quad \forall i,\ \forall(i,v)\in\mathcal{E}.$$

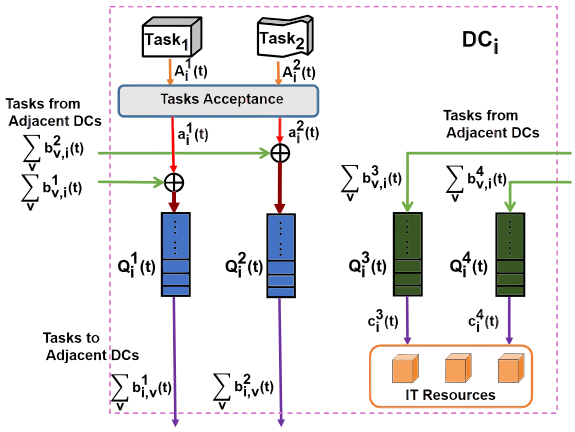

Fig. 2. *GlobalAny* scenario for scheduling data-oriented tasks in *DC* i.

Then, for queue $Q_i^k(t),\ \forall k\notin\mathcal{K}_i$ in *DC* i, we insert both the locally-accepted and remotely-sent-over tasks (all belonging to the k-th category), while the sojourn tasks are removed and sent to the next hop. Since the storage space in a DC is usually big enough, we can treat the queues as infinite ones. Hence, $Q_i^k(t)$ for task acceptance and transfer is updated as

$$\begin{aligned} Q_i^k(t+1)=&\max\left(Q_i^k(t)-\sum_{\{v:(i,v)\in\mathcal{E}\}}b_{i,v}^k(t),\ 0\right)\\ &+\sum_{\{v:(v,i)\in\mathcal{E}\}}b_{v,i}^k(t)+a_i^k(t)\cdot\delta_k,\quad \forall i,\ k\notin\mathcal{K}_i. \end{aligned} \tag{5}$$

3) Task Processing: When a task arrives at one of its feasible destination DCs, it is inserted into the queue for final processing and will not be sent to other DCs anymore. Then, the destination DC allocates IT resources to process the task. We denote the allocated resources to the k-th category tasks in *DC* i as $c_i^k(t)$. Apparently, the total amount of IT resources should not exceed those available in each DC, as

$$\sum_k c_i^k(t)\le C_i,\quad \forall i. \tag{6}$$

Similarly, we denote the time-average expectation of $c_i^k(t)$ as c_i^k. Then, as for queue $Q_i^k(t),\ \forall k\in\mathcal{K}_i$ in *DC* i, the remotely-sent-over tasks are added and the sojourn requests are removed for final processing. We use μ_k to denote the number of k-th category tasks that unit IT resource can process per TS. Hence, we can update the queue for task processing as

$$\begin{aligned} Q_i^k(t+1)=&\max\left(Q_i^k(t)-c_i^k(t)\cdot\mu_k\cdot\delta_k,\ 0\right)\\ &+\sum_{\{v:(v,i)\in\mathcal{E}\}}b_{v,i}^k(t),\quad \forall i,\ k\in\mathcal{K}_i. \end{aligned} \tag{7}$$

Fig. 2 gives an example to illustrate the *GlobalAny* scheduling scheme. We assume that there are four categories of tasks running in the cloud DCs. *DC* i is the source DC of tasks belonging to first two categories, and it is also a destination DC of the tasks in the last two categories, *i.e.*, $\mathcal{K}_i=\{3,4\}$. Firstly, the tasks generated as ones in the first two categories will be accepted or dropped. Then, the accepted ones and tasks from adjacent DCs will be inserted into the corresponding queue $Q_i^k(t)$. If the tasks can be processed in *DC* i (*i.e.*, they are

in the last two categories), we should allocate IT resources to process them. Otherwise, we should transfer the tasks to the adjacent DCs. And all the tasks are transferred according to this strategy until they arrive at one of their destination DCs. In addition, Eqs. (5) and (7) indicate that the more bandwidth resources and IT resources be allocated, the faster the tasks can be transferred and processed.

C. Profit-Driven Optimization Model

The discussion above indicates that the data-oriented tasks consume both bandwidth resources in the network and IT resources in the DCs and there is a tradeoff between the tasks processing latency and resource consumption. Since the customers in the multi-DC system connect to their source DCs and can submit data-oriented tasks there, we formulate a profit model by considering the revenue from serving the tasks and the cost due to the usages of bandwidth and IT resources. The revenue is calculated based on two payments. The CSP gets the first payment from the customer who submits the task, when the corresponding source DC accepts the task. And the second payment comes in (*i.e.*, also from the customer) when the task's data has been processed by its destination DC. As the DCs handle task acceptance and processing separately, the revenue model addresses the two important milestones of serving a data-oriented task in the multi-DC system with the two payments. Apparently, the revenue from the first payment is related to the time-average expectation of task acceptance rates, *i.e.*, $\{a_i^k,\ \forall i,k\}$. Here, we use a logarithmic utility model, which follows the law of diminishing marginal utility [34] and is widely used in previous work.

$$f_1(a_i^k) = log(1+\alpha_k \cdot a_i^k), \quad \forall i,k, \tag{8}$$

where α_k is the revenue coefficient for the k-th task category. The second payment is based on the number of tasks that have been processed. At time t, the number of processed k-th category tasks is $m_k(t)$, as

$$m_k(t) = \sum_i c_i^k(t) \cdot \mu_k, \quad \forall k. \tag{9}$$

The time-average expectation of $m_k(t)$ is m_k. Then, with the linear utility model in [38], we get this part of revenue as

$$f_2(m_k) = \gamma_k \cdot m_k, \quad \forall k, \tag{10}$$

where γ_k is the coefficient for tasks in the k-th category.

The cost of the bandwidth usage on link $(i,v) \in \mathcal{E}$ is

$$h_1(b_{i,v}) = \beta_{i,v} \cdot b_{i,v}, \quad \forall (i,v) \in \mathcal{E}, \tag{11}$$

where $\beta_{i,v}$ is the cost of unit bandwidth usage on link (i,v). The last part of cost comes from the IT resource usage due to the store-and-forward operations. At time t, the amount of data that arrives at an intermediate *DC* i and will be forwarded to a remote DC later is $n_i(t)$, as

$$n_i(t) = \sum_{k,v:(v,i)\in\mathcal{E}} b_{v,i}^k(t) \cdot \mathcal{I}_{\{k \notin \mathcal{K}_i\}}, \quad \forall i, \tag{12}$$

and the time-average expectation of $n_i(t)$ is n_i. Then, the time-average cost due to IT resource usage in *DC* i is

$$h_2(n_i) = \rho_i \cdot n_i, \quad \forall i, \tag{13}$$

where ρ_i is the cost coefficient of the IT resources used by the store-and-forward operations in *DC* i.

Finally, the time-average profit from serving the tasks can be obtained as

$$P = \sum_{i,k} f_1(a_i^k) + \sum_k f_2(m_k) - \sum_{i,v} h_1(b_{i,v}) - \sum_i h_2(n_i). \tag{14}$$

Here, we try to maximize the time-average profit while all the constraints should be satisfied. In addition, all the queues in the system should keep stable, otherwise the processing latency of accepted tasks may be infinite [14]. Hence, the profit-driven optimization is modeled as

$$\begin{aligned} &Maximize \quad P, \\ &s.t. \quad \text{Eqs. (1) - (7)}, \\ &\qquad Q_i^k(t) \text{ keeps stable}, \quad \forall i,k. \end{aligned} \tag{15}$$

IV. *GlobalAny* Scheduling Algorithm

A. Lyapunov Optimization Techniques

We use the Lyapunov optimization techniques to solve the optimization in Eq. (15) and develop a profit-driven distributed online scheduling algorithm. As the utility function $f_1(\cdot)$ is nonlinear, we first introduce an auxiliary variable, $x_i^k(t)$, to transform the original problem into the standard drift-minus-profit framework in Lyapunov optimization, as

$$0 \le x_i^k(t) \le A_k^{max} \cdot \mathcal{I}_{\{i\in\mathcal{S}_k\}}, \quad \forall i,k. \tag{16}$$

And we denote the time-average expectation of $x_i^k(t)$ as $\overline{x_i^k}$. Then, the original problem is transformed into

$$\begin{aligned} Maximize \quad & \sum_{i,k} \overline{f_1(x_i^k)} + \sum_k f_2(m_k) \\ & - \sum_{i,v} h_1(b_{i,v}) - \sum_i h_2(n_i), \\ s.t. \quad & \text{Eqs. (1) - (7)}, \\ & x_i^k \le a_i^k, \quad \forall i,k, \\ & Q_i^k(t) \text{ keeps stable}, \quad \forall i,k, \end{aligned} \tag{17}$$

where we have

$$\overline{f_1(x_i^k)} = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{log(1+\alpha_k \cdot x_i^k(\tau))\}, \quad \forall i,k.$$

Then, we use the virtual queues, $\{X_i^k(t),\ \forall i,k\}$ with initial values as $X_i^k(0) = 0$, to transform the constraint

$$x_i^k \le a_i^k, \quad \forall i,k,$$

in Eq. (17) into a queue-stable problem [14]. Similar to $\boldsymbol{Q}(t)$ in Eq. (2), we use $\boldsymbol{X}(t)$ to denote the queue matrix for $\{X_i^k(t),\ \forall i,k\}$, and $X_i^k(t)$ is updated as

$$X_i^k(t+1) = \max(X_i^k(t) - a_i^k(t) + x_i^k(t), 0) \quad \forall i,k. \tag{18}$$

Let $\boldsymbol{\Omega}(t) = [\boldsymbol{Q}(t), \boldsymbol{X}(t)]$, and we define the Lyapunov function $\mathbb{L}(\boldsymbol{\Omega}(t))$ as

$$\mathbb{L}(\boldsymbol{\Omega}(t)) = \frac{1}{2}\left\{\sum_{i,k}(Q_i^k(t)\cdot w_k)^2 + \sum_{i,k}(X_i^k(t)\cdot \eta_k)^2\right\}, \tag{19}$$

The Lyapunov drift function is the conditional expectation of the Lyapunov function in Eq. (19) for different TS', as

$$\Delta(\boldsymbol{\Omega}(t)) = \mathbb{E}\{\mathbb{L}(\boldsymbol{\Omega}(t+1)) - \mathbb{L}(\boldsymbol{\Omega}(t)) \mid \boldsymbol{\Omega}(t)\}. \tag{20}$$

If we define $\zeta(t)$ as

$$\zeta(t) = \sum_{i,k} f_1(x_i^k(t)) + \sum_k f_2(m_k(t)) - \sum_{i,v} h_1(b_{i,v}(t)) - \sum_i h_2(n_i(t)),$$

the drift-minus-profit expression can be obtained as

$$\Phi(\mathbf{\Omega}(t)) = \Delta(\mathbf{\Omega}(t)) - V \cdot \mathbb{E}\{\zeta(t) \mid \mathbf{\Omega}(t)\}, \tag{21}$$

where V is the tradeoff parameter to balance the queue lengths and the time-average profit. Eq. (21) satisfies the inequality below because $[\max(q-c,0)+a]^2 \leq q^2+a^2+c^2+2q(a-c)$.

$$\Phi(\mathbf{\Omega}(t)) \leq B + \Phi_1(\mathbf{\Omega}(t)) + \Phi_2(\mathbf{\Omega}(t)) + \Phi_3(\mathbf{\Omega}(t)) + \Phi_4(\mathbf{\Omega}(t)), \tag{22}$$

where B is a constant that satisfies

$$\begin{aligned} B \geq \frac{1}{2} \cdot \sum_{i,k} \mathbb{E}\Bigg\{ & \left[a_i^k(t) - x_i^k(t)\right]^2 \cdot \eta_k^2 \\ & + \left[(\sum_v b_{v,i}^k(t) + a_i^k(t) \cdot \delta_k)^2 + (\sum_v b_{i,v}^k(t))^2 \right] \cdot w_k^2 \cdot \mathcal{I}_{\{k \notin \mathcal{K}_i\}} \\ & + \left[(\sum_v b_{v,i}^k(t))^2 + (c_i^k(t) \cdot \mu_k \cdot \delta_k)^2 \right] \cdot w_k^2 \cdot \mathcal{I}_{\{k \in \mathcal{K}_i\}} \mid \mathbf{\Omega}(t) \Bigg\}. \end{aligned}$$

And $\Phi_1(\mathbf{\Omega}(t))$, $\Phi_2(\mathbf{\Omega}(t))$, $\Phi_3(\mathbf{\Omega}(t))$ and $\Phi_4(\mathbf{\Omega}(t))$ have the expressions as

$$\Phi_1(\mathbf{\Omega}(t)) = \sum_{i,k} \mathbb{E}\{X_i^k(t) \cdot \eta_k^2 \cdot x_i^k(t) - V \cdot f_1(x_i^k(t)) \mid \mathbf{\Omega}(t)\}, \tag{23}$$

$$\begin{aligned} \Phi_2(\mathbf{\Omega}(t)) = \sum_{i,k} \mathbb{E}\{ & Q_i^k(t) \cdot \delta_k \cdot w_k^2 \cdot a_i^k(t) \\ & - X_i^k(t) \cdot \eta_k^2 \cdot a_i^k(t) \mid \mathbf{\Omega}(t)\}, \end{aligned} \tag{24}$$

$$\begin{aligned} \Phi_3(\mathbf{\Omega}(t)) = \sum_{i,v,k} \mathbb{E}\{ & Q_i^k(t) \cdot (b_{v,i}^k(t) - b_{i,v}^k(t)) \cdot w_k^2 \\ & + V \cdot (\beta_{i,v} + \rho_v \cdot \mathcal{I}_{\{k \notin \mathcal{K}_v\}})) \cdot b_{i,v}^k(t) | \mathbf{\Omega}(t)\}. \end{aligned} \tag{25}$$

$$\begin{aligned} \Phi_4(\mathbf{\Omega}(t)) = \sum_{i,k} \mathbb{E}\{ & (- Q_i^k(t) \cdot w_k^2 \cdot c_i^k(t) \cdot \mu_k \cdot \delta_k \\ & - V \cdot c_i^k(t) \cdot \gamma_k \cdot \mu_k) \cdot \mathcal{I}_{\{k \in \mathcal{K}_v\}} \mid \mathbf{\Omega}(t)\}. \end{aligned} \tag{26}$$

B. Distributed Online Scheduling

We design a scheduling algorithm (*i.e.*, *GlobalAny*) to handle the auxiliary variables, tasks' acceptance, data-transfers and processing in an online and distributed manner. We will also verify that only small communication overhead is needed for information exchange.

1) Auxiliary Variables: We can get the optimal solutions of $\{x_i^k(t),\ \forall i,k,t\}$ by minimizing Eq. (23). For these independent variables, we have

$$\begin{aligned} Minimize \quad & X_i^k(t) \cdot \eta_k^2 \cdot x_i^k(t) - V \cdot log(1 + \alpha_k \cdot x_i^k(t)) \\ s.t. \quad & 0 \leq x_i^k(t) \leq A_k^{max}, \quad \forall i,k, \end{aligned} \tag{27}$$

and the optimal solutions of $\{x_i^k(t),\ \forall i,k,t\}$ are

$$x_i^k(t)^* = \begin{cases} 0, & X_i^k(t) > \frac{V \cdot \alpha_k}{\eta_k^2}, \\ \frac{V}{X_i^k(t) \cdot \eta_k^2} - \frac{1}{\alpha_k}, & X_i^k(t) \in \left[\frac{V \cdot \alpha_k}{(1 + \alpha_k \cdot A_k^{max}) \cdot \eta_k^2},\ \frac{V \cdot \alpha_k}{\eta_k^2} \right], \\ A_k^{max}, & X_i^k(t) \in \left[0,\ \frac{V \cdot \alpha_k}{(1 + \alpha_k \cdot A_k^{max}) \cdot \eta_k^2}\right). \end{cases} \tag{28}$$

2) Task Acceptance Policy: When the real queue is not longer than the virtual queue, *i.e.*, $Q_i^k(t) \leq X_i^k(t) \cdot \delta_k$, *GlobalAny* should accept all the newly-generated k-th category tasks $A_i^k(t)$ in *DC* i at time t, otherwise, it should drop all of them. This is because we can get the optimal task acceptance policy by minimizing Eq. (24), as

$$\begin{aligned} Minimize \quad & Q_i^k(t) \cdot \delta_k \cdot w_k^2 \cdot a_i^k(t) - X_i^k(t) \cdot \eta_k^2 \cdot a_i^k(t), \\ s.t. \quad & 0 \leq a_i^k(t) \leq A_i^k(t) \cdot \mathcal{I}_{\{i \in \mathcal{S}_k\}}, \quad \forall i,k, \end{aligned} \tag{29}$$

and thus the optimal solution $a_i^k(t)^*$ is

$$a_i^k(t)^* = \begin{cases} A_i^k(t), & Q_i^k(t) \leq X_i^k(t) \cdot \delta_k, \\ 0, & \text{otherwise.} \end{cases} \tag{30}$$

3) Bandwidth Allocation: To get the optimal bandwidth allocation, we first define $B_{i,v}^k(t)$ as

$$\begin{aligned} B_{i,v}^k(t) = & (Q_i^k(t) - Q_v^k(t)) \cdot w_k^2 \\ & - V \cdot (\beta_{i,v} + \rho_v \cdot \mathcal{I}_{\{k \notin \mathcal{K}_v\}}), \quad \forall i,v,k. \end{aligned} \tag{31}$$

Then, we can conclude that if the queue lengths are not considered, *GlobalAny* should provide all the bandwidth on link (i,v) to the queue that has the maximum positive $B_{i,v}^k(t)$ at time t. This is because the optimal bandwidth allocation can be obtained by minimizing $\Phi_3(\mathbf{\Omega}(t))$ in Eq. (25). If the queue lengths are not considered, we can assume that the bandwidth allocation would not result in empty queues, *i.e.*, there are always enough data-transfers on each link. Hence, Eq. (25) is independent of link status and can be transformed into

$$\begin{aligned} Maximize \quad & \sum_k B_{i,v}^k(t) \cdot b_{i,v}^k(t) \\ s.t. \quad & 0 \leq \sum_k b_{i,v}^k(t) \leq B_{i,v}, \quad \forall (i,v) \in \mathcal{E}, \end{aligned} \tag{32}$$

and we can get the optimal $b_{i,v}^k(t)^*$ as follows.

$$b_{i,v}^k(t)^* = \begin{cases} B_{i,v}(t), & k = k^*, \\ 0, & \text{otherwise}, \end{cases} \tag{33}$$

where we have

$$k^* = \text{argmax}\{B_{i,v}^k(t) | B_{i,v}^k(t) > 0, \forall k \in \mathcal{K}\}.$$

Therefore, we verify that in DC i, *GlobeAny* should allocate the bandwidth on each link to the queue that has the maximum positive $B_{i,v}^k(t)$. Note that, $B_{i,v}^k(t)$ equals to the weighted differential queue length (*i.e.*, $(Q_i^k(t) - Q_v^k(t)) \cdot w_k^2$) subtracting a fixed value, which is determined by the cost of data-transfer using store-and-forward. As *GlobalAny* allocates the bandwidth to the task category that has maximum weighted differential queue length, we can provide differentiated services to the task categories by adjusting their weights $\{w_k\}$.

However, the analysis above overlooks the scenario that the queue with the maximum positive $B_{i,v}^k(t)$ does not have enough data to fully utilize the allocated bandwidth. For this scenario, the bandwidth allocation according to the aforementioned scheme becomes inefficient since bandwidth could be wasted. Hence, we introduce a new constraint to make sure that all the allocated bandwidth is fully utilized in each DC.

$$0 \leq \sum_{v:(i,v)\in\mathcal{E}} b_{i,v}^k(t) \leq Q_i^k(t), \quad \forall i,k. \tag{34}$$

Since each data-oriented task will be forwarded to one and only one destination DC, the allocated bandwidth should be in integer times of the data-size of each task category. Thus, we introduce an integer variable $N_{i,v}^k$ to represent the number of tasks can be transferred on link (i,v) and we have

$$b_{i,v}^k(t) = N_{i,v}^k \cdot \delta_k, \quad \forall i,k,\ \forall (i,v)\in\mathcal{E}. \tag{35}$$

Then, we design a new optimization model for the data-transfer in each DC as follows.

$$\begin{aligned} Maximize\ & \sum_{v,k} B_{i,v}^k(t)\cdot b_{i,v}^k(t) \\ s.t.\ & 0 \leq \sum_k b_{i,v}^k(t) \leq B_{i,v},\ \forall (i,v)\in\mathcal{E}, \\ & 0 \leq \sum_{v:(i,v)\in\mathcal{E}} b_{i,v}^k(t) \leq Q_i^k(t),\ \forall k, \\ & b_{i,v}^k(t) = N_{i,v}^k\cdot\delta_k,\ \forall k,(i,v)\in\mathcal{E}. \end{aligned} \tag{36}$$

And the optimization in Eq. (36) can be rewritten as follows if we notice the relation between $b_{i,v}^k(t)$ and $N_{i,v}^k$.

$$\begin{aligned} Maximize\ & \sum_{v,k} B_{i,v}^k(t)\cdot\delta_k\cdot N_{i,v}^k \\ s.t.\ & 0 \leq \sum_k N_{i,v}^k\cdot\delta_k \leq B_{i,v},\ \forall (i,v)\in\mathcal{E}, \\ & 0 \leq \sum_{v:(i,v)\in\mathcal{E}} N_{i,v}^k\cdot\delta_k \leq Q_i^k(t),\ \forall k. \end{aligned} \tag{37}$$

Note that, the optimization in Eq. (37) represents a bounded multiple knapsack problem, which is known to be $\mathcal{NP}$-hard [39]. Therefore, we design an approximation algorithm that leverages the linear program relaxation (LP-relaxation) with rounding, as shown in *Algorithm* 1. Specifically, in each DC, we first solve the problem in Eq. (37) by relaxing the integer variables and then round down the solution to get $\{N_{i,v}^k,\ \forall v,k\}$. Then, we allocate the residual bandwidth $B_{i,v}$ on each link (i,v) to the remaining data in $\{Q_i^k(t),\ \forall k\}$. This means that we sort the combinations of $\{v,k\}$ in descending order of $B_{i,v}^k(t)\cdot\delta_k$, and then for each sorted $\{v,k\}$, we allocate the bandwidth on link (i,v) to queue $Q_i^k(t)$ accordingly. Finally, the bandwidth allocation $b_{i,v}^k(t)$ is obtained.

4) Task Processing: To get the optimal IT resource allocation for task processing in destination DCs, we define

$$C_i^k(t) = (Q_i^k(t)\cdot w_k^2\cdot\delta_k + V\cdot\gamma_k)\cdot\mu_k, \quad \forall i,k\in\mathcal{K}_i.$$

Then, *GlobalAny* should allocate IT resources in destination DCs for task processing according to $C_i^k(t)$. This is because according to Eq. (26), the IT resource allocation can be obtained by solving the optimization problem below

$$\begin{aligned} Maximize\quad & \sum_{k\in\mathcal{K}_i} C_i^k(t)\cdot c_i^k(t), \\ s.t.\quad & \sum_{k\in\mathcal{K}_i} c_i^k(t) \leq C_i,\ \forall i. \end{aligned} \tag{38}$$

Since $C_i^k(t)\geq 0,\ \forall i,k\in\mathcal{K}_i$ is always true, we can get the optimal $c_i^k(t)^*$ with the following procedure. Firstly, in *DC* i, we sort the queues $\{Q_i^k(t),\ \forall k\in\mathcal{K}_i\}$ in descending order of $C_i^k(t)$. Secondly, we allocate IT resources to them in sorted order, until they all get sufficient IT resources to handle the tasks or all the IT resources are allocated.

5) Overall Distributed Online Scheduling: Based on the analysis above, we get the overall procedure of *GlobalAny* for scheduling data-oriented tasks at time t in *DC* i, which is shown in *Algorithm* 2. Specifically, for each task category, the auxiliary variable and task acceptance are first calculated independently with Eqs. (28) and (30), respectively. Then, we use *Algorithm* 1 to get the bandwidth allocation and forward the related tasks to adjacent DCs. Meanwhile, all the tasks received from adjacent DCs should be accepted. Next, *GlobalAny* handles the tasks whose destination DCs are *DC* i with the scheme proposed in Section IV-B4. Finally, we update $\{Q_i^k(t)\}$ and $\{X_i^k(t)\}$ with Eqs. (5), (7) and (18), and share the queue lengths of $\{Q_i^k(t)\}$ with adjacent DCs.

We should point out that *GlobalAny* realizes the inter-DC data-oriented tasks transferring and processing in a distributed way. Basically, all the accepted tasks will be delivered with the store-and-forward scheme in *Algorithm* 1, until they reach any of their destination DCs. Then, each task's data will be stored and processed. Meanwhile, all the DCs should exchange the information on queue lengths to their adjacent DCs with *Algorithm* 1 in each TS. Note that, the bandwidth overhead for

Algorithm 1 Bandwidth Allocation of *GlobalAny* for *DC* i

Input:
$\mathbf{Q}(t),\ \{B_{i,v},\ \forall v\}$.

calculate $\{B_{i,v}^k(t),\ \forall v,k\}$ with Eq. (31);
get $\{N_{i,v}^k,\ \forall v,k\}$ with LP-relaxation of Eq. (37);
$N_{i,v}^k = \lfloor N_{i,v}^k \rfloor,\ \forall v,k$;
$Q_i^k(t) = Q_i^k(t) - \sum_v N_{i,v}^k\cdot\delta_k,\ \forall k$;
$B_{i,v} = B_{i,v} - \sum_k N_{i,v}^k\cdot\delta_k,\ \forall v$;
sort $\{v,k\}$ in descending order of $B_{i,v}^k(t)\cdot\delta_k$;
for each $\{v,k\}$ in sorted order **do**
  if $B_{i,v}^k(t)\geq 0$ **then**
    $\Delta = \lfloor \frac{\max(\min(Q_i^k(t),B_{i,v}),0)}{\delta_k} \rfloor$;
    $N_{i,v}^k = N_{i,v}^k + \Delta$;
    $Q_i^k(t) = Q_i^k(t) - \Delta\cdot\delta_k$;
    $B_{i,v} = B_{i,v} - \Delta\cdot\delta_k$;
  else
    break;
  end if
end for
$b_{i,v}^k(t) = N_{i,v}^k\cdot\delta_k,\ \forall v,k$;

Algorithm 2 Tasks Scheduling in *DC* i using *GlobalAny*

1: **for** each task category $k \in \mathcal{K}$ **do**
2: calculate auxiliary variable $x_i^k(t)$ with Eq. (28);
3: accept tasks according to $x_i^k(t)$ using Eq. (30);
4: **end for**
5: use *Algorithm* 1 to obtain the bandwidth allocation;
6: send tasks in queues to adjacent DCs with the allocated bandwidth;
7: accept all the tasks from adjacent DCs;
8: process the tasks whose destination DC set includes *DC* i with the scheme proposed in Section IV-B4;
9: update $\{Q_i^k(t),\ X_i^k(t),\ \forall k\}$;
10: send queue lengths of $\{Q_i^k(t), \forall k\}$ to adjacent DCs;

the information exchange is negligible, when compared with the bandwidth used for the data-transfer.

C. Optimality Analysis

With the procedure demonstrated in [14, 34], we can easily prove that the time-average profit P obtained by *GlobalAny* can approach to the theoretical maximum profit P^* arbitrarily within a constant gap $O(\frac{1}{V})$.

V. Data-Transfer Acceleration

GlobalAny can schedule the data-oriented tasks in the cloud DC system to achieve the profit that is arbitrarily close to the optimal one. However, since we do not put queue lengths in the optimization objective, the resulting data-transfer latency could become an issue. Specifically, the store-and-forward scheme and hop-by-hop bandwidth allocation governed by *Algorithm* 1 could induce relatively long latency. For instance, in Fig. 3, the data in *DC* s needs to be sent to *DC* d. Then, the data-transfer will take at least 3 TS' if the data is forwarded hop-by-hop on s-u-v-d. Moreover, store-and-forward might cause routing loops, *i.e.*, data can be forwarded back and forth in a loop before reaching its destination DC. Note that, the latency can be reduced, if we modify the bandwidth allocation scheme to bypass certain intermediate DC(s). Hence, we design a data-transfer acceleration scheme for this purpose.

The basic idea of the data-transfer acceleration is that in each TS, we make the DCs calculate bandwidth allocation collaboratively in multiple iterations to determine a forwarding path segment instead of the next hop for each data-oriented task. For instance, in Fig. 3, we make *DCs* s, u and v calculate the bandwidth allocation scheme in three iterations to set up the direct-transfer path s-u-v-d. Firstly, *DC* s determines the bandwidth allocation on link (s, u) and forwards the related information to *DC* u. Note that here, *DC* s only lets *DC* u know that the data will go through it, but the actual data still stays in *DC* s. Then, *DC* u updates its queue as it has already received the data, obtains the bandwidth allocation on link (u, v), and sends the related information to *DC* v. *DC* v repeats the same procedure to get the bandwidth allocation on link (v, d). All these operations are performed within one TS, and then we start the actual data-transfer. Note that when the data is sent out from *DC* s, each intermediate DC on s-u-v-d checks

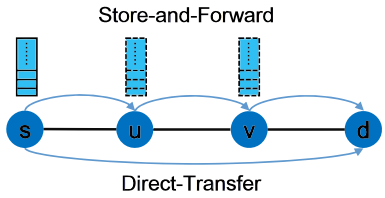

Fig. 3. Example on data-transfer acceleration.

whether the bandwidth has already been allocated for it to the next hop, if yes, the data is sent out directly, otherwise, it is buffered in the intermediate DC and waits for the next TS.

Algorithm 3 shows the data-transfer acceleration scheme designed for *GlobalAny*. *Lines* 1-2 are for the initialization. Here, we use flag $F_{i,v}^k = 1$ to indicate that the data-transfer of the k-th task category can use link (i, v), and $F_{i,v}^k = 0$ otherwise. Hence, reverse data-transfer can be avoided to minimize the probability of forming routing loops. Specifically, if a link is used by the data-transfer of a task category, we forbid the data-transfer in the reverse direction in the subsequent iterations by setting the corresponding $F_{i,v}^k = 0$. Hence, we have

$$0 \le N_{i,v}^k \cdot \delta_k \le Q_i^k(t) \cdot F_{i,v}^k, \quad \forall i, k,\ \forall (i, v) \in \mathcal{E}. \tag{39}$$

In each iteration of the while-loop that covers *Lines* 3-18, DC i tries to extend the forwarding path segments to one more hop. Here, due to the time constraint from TS on the information exchanges among the DCs, we assume that each DC can only extend the forwarding path segments to M hops at most within one TS. In *Lines* 5-13, *DC* i tries to determine the new bandwidth allocation on each link that origins from it. Specifically, *Line* 5 calculates the bandwidth allocation by using LP-relaxation to solve the problem of Eq. (37) with the constraint in Eq. (39). And we also forbid the data-transfers in the reverse direction in the subsequent iterations. *Lines* 12-13 update the corresponding variables. We perform message exchanges among DCs and update other variables in *Lines* 14-17. Although the data-transfer acceleration increases the information exchanges in *GlobalAny*, each DC still only needs to exchange information with its adjacent DCs. And at last, we perform the residual bandwidth allocation for the remaining data in *Lines* 19-29, which is similar to *GlobalAny* algorithm. We denote the algorithm that incorporates the data-transfer acceleration scheme as *GlobalAny_Ext*.

VI. Performance Evaluation

Numerical simulations with two different topologies are performed to evaluate our proposed algorithms. One is the full-mesh 7-node Amazon EC2 network topology in [40]. We adopt the same link bandwidth and scale down the link cost with 10^{-5} proportionally for normalization. The other is the 12-node B4 topology in [12]. The link available bandwidth is randomly selected within $[10, 100]$ Gb/s, whose unit cost is $[1, 10] \times 10^{-2}$ per Gb/s. The number of task categories, *i.e.*, $|\mathcal{K}|$, is set as 100. The arrivals of the data-oriented tasks follow the uniform distribution, and the time average arrival rate is half of the maximum value, *i.e.*, $\frac{1}{2}A_k^{max}$. The data-size

Algorithm 3 *GlobalAny* with Data-Transfer Acceleration

Input:
$\mathbf{Q}(t),\ \{B_{i,v}\},\ \{B^k_{i,v}(t)\}$.

$F^k_{i,v} = 1,\ b^k_{i,v}(t) = 0,\ \forall i, v, k,\ n = 0$;
while $n < M$ **do**
  $n = n + 1$;
  get $\{N^k_{i,v},\ \forall v, k\}$ with LP-relaxation of Eq. (37) and the constraint in Eq. (39);
  $N^k_{i,v} = \lfloor N^k_{i,v} \rfloor,\ \forall v, k$;
  for each $\{v, k\}$ **do**
    if $N^k_{i,v} > 0$ **then**
      $F^k_{v,i} = 0$;
    end if
  end for
  $b^k_{i,v}(t) = b^k_{i,v}(t) + N^k_{i,v} \cdot \delta_k,\ \forall v, k$;
  $B_{i,v} = B_{i,v} - \sum_k N^k_{i,v} \cdot \delta_k,\ \forall v$;
  exchange $F^k_{i,v}$, $F^k_{v,i}$ and $N^k_{i,v}$ with adjacent DCs;
  $Q^k_i(t) = Q^k_i(t) - (\sum_v N^k_{i,v} + \sum_v N^k_{v,i}) \cdot \delta_k,\ \forall k$;
  exchange queue lengths of $\mathbf{Q}(t)$ with adjacent DCs;
  update $B^k_{i,v}(t)$ with Eq. (31);
end while
sort $\{v, k\}$ in descending order of $B^k_{i,v}(t) \cdot \delta_k$;
for each $\{v, k\}$ in sorted order **do**
  if $B^k_{i,v}(t) \geq 0$ **then**
    $\Delta = \lfloor \frac{\max(\min(Q^k_i(t), B_{i,v}), 0)}{\delta_k} \rfloor$;
    $b^k_{i,v}(t) = b^k_{i,v}(t) + \Delta \cdot \delta_k$;
    $Q^k_i(t) = Q^k_i(t) - \Delta \cdot \delta_k$;
    $B_{i,v} = B_{i,v} - \Delta \cdot \delta_k$;
  else
    break;
  end if
end for

of each category δ_k is randomly selected within $[1, 10]$ M-bits. The source and destination DC sets of each category are also randomly selected and do not overlap. The IT resources in each DC are set as 10^4, and the number of k-th category tasks that can be processed with a unit IT resource in one TS, *i.e.*, μ_k, is selected within $[1, 10]$.

TABLE II
SIMULATION PARAMETERS

$\lvert\mathcal{K}\rvert$, number of task categories	100
δ_k, data-size of k-th category tasks	$[1, 10]$ M-bits
α_k, revenue coefficient of k-th task category	$[0.01, 0.1]$
γ_k, revenue coefficient of *DC* i	$[1, 2] \times 10^{-3}$
ρ_i, cost coefficient of *DC* i	$[1, 2] \times 10^{-4}$
w_k, weight of k-th task category	$[0.1, 1]$
A^{max}_k, maximum arrival-rate of k-th category	100
C_i, the amount of IT resources in *DC* i	10^4
μ_k, number of k-th category tasks that unit IT resource can process in one TS	$[1, 10]$
Δt, duration of one TS	1 minute
TS' in each simulation	10000

For the data-transfer acceleration, we determine M as the diameter of the topology, *i.e.*, the maximum hop-count of the shortest paths between DC-pairs in the network. Hence, *GlobalAny_Ext* becomes *GlobalAny* as $M = 1$ in the EC2 topology. For the B4 topology, we have $M = 5$. TS is set as 1 minute and we simulate 10,000 TS' in each simulation. Table II summarizes all the key simulation parameters. In the simulations, we introduce two performance metrics, *i.e.*, the time-average delay and throughput. Here, we adopt D to denote the average delay of all the processed tasks, and represent the time-average throughput, which is the average amount of processed data per TS in the network, as T.

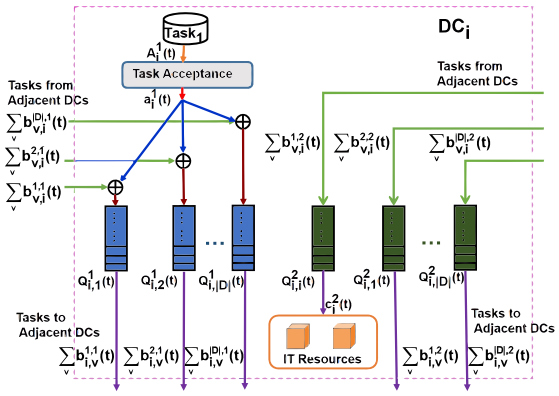

Fig. 4. *LocalAny* scenario for scheduling data-oriented tasks in *DC* i.

A. Benchmark Algorithms

1) LocalAny: A more straight-forward idea to schedule the data-oriented tasks is that when a task is generated, its source DC decides whether to serve it or not and where its destination DC should be in one shot. As the destination DCs are selected locally in source DCs, we name this scheduling algorithm as *LocalAny*. Here, for *LocalAny*, $Q^k_{i,j}(t)$ is used to denote the queue that stores all the tasks that belong to the k-th category and have *DC* j as the destination DC. In *DC* i, we use $a^k_{i,j}(t)$ to denote the accepted tasks that have *DC* j as the destination DC, and $b^{j,k}_{i,v}(t)$ to denote the allocated bandwidth on link (i, v) for the tasks that target to *DC* j, respectively, while $c^k_i(t)$ is the allocated IT resources for task processing. Then, the queues for task acceptance and transfer are updated as

$$\begin{aligned} Q^k_{i,j}(t+1) = \max\Bigg(& Q^k_{i,j}(t) - \sum_{\{v:(i,v)\in\mathcal{E}\}} b^{j,k}_{i,v}(t),\ 0 \Bigg) \\ & + \sum_{\{v:(v,i)\in\mathcal{E}\}} b^{j,k}_{v,i}(t) + a^k_{i,j}(t) \cdot \delta_k, \\ & \{i, j, k : i, j \in \mathcal{D}, i \neq j, k \notin \mathcal{K}_i\}. \end{aligned} \tag{40}$$

The queues for task processing are updated as

$$\begin{aligned} Q^k_{i,i}(t+1) = \max & \left(Q^k_{i,i}(t) - c^k_i(t) \cdot \mu_k,\ 0 \right) \\ & + \sum_{\{v:(v,i)\in\mathcal{E}\}} b^{i,k}_{v,i}(t),\ \forall i \in \mathcal{D}, k \in \mathcal{K}_i. \end{aligned} \tag{41}$$

Fig. 4 shows an example on the data-transfer with *LocalAny*, where for simplicity, we only include two task categories. We assume that *DC* i is the source DC of tasks in the first category and it is the destination DC of the tasks in the second category, *i.e.*, $i \in \mathcal{S}_1$ and $i \in \mathcal{D}_2$. The destination DCs of the tasks are

determined when they are accepted. Then, the newly accepted tasks and those from adjacent DCs will be inserted into the corresponding queues. We allocate IT resources to the tasks that arrive at their pre-selected destination DC, *e.g.*, the tasks in $Q^2_{i,i}(t)$ in Fig. 4. And the bandwidth allocation for other queues is performed afterwards.

With the Lyapunov optimization techniques, we get the optimization for the task acceptance in *LocalAny* as

$$\begin{aligned} Minimize \quad & \sum_j Q^k_{i,j}(t)\cdot a^k_{i,j}(t) - X^k_i(t)\cdot\delta_k\cdot a^k_i(t), \\ s.t. \quad & 0 \le a^k_i(t) \le A^k_i(t),\ \forall i,k, \\ & \sum_j a^k_{i,j}(t) = a^k_i(t),\ \forall i,k. \end{aligned} \tag{42}$$

Here, $\{X^k_i(t), \forall i,k\}$ are still the virtual queues. Supposing that we have already got the optimal solution of $a^k_i(t)$ as $a^k_i(t)^*$, we can reduce the optimization to

$$\begin{aligned} Minimize \quad & \sum_j Q^k_{i,j}(t)\cdot a^k_{i,j}(t), \\ s.t. \quad & \sum_j a^k_{i,j}(t) = a^k_i(t),\ \forall i,k. \end{aligned} \tag{43}$$

And $a^k_{i,j}(t)^*$ can be obtained as

$$a^k_{i,j}(t)^* = \begin{cases} a^k_i(t)^*, & j = argmin\{Q^k_{i,v}(t), \forall v \in \mathcal{D}_k\}, \\ 0, & \text{otherwise}, \end{cases} \tag{44}$$

which means that all the newly-generated requests should be allocated to the shortest feasible queue. Then, we put the optimal solution $a^k_{i,j}(t)^*$ into Eq. (42), and get

$$\begin{aligned} Minimize \quad & (Q^k_{i,j^*}(t) - X^k_i(t)\cdot\delta_k)\cdot a^k_i(t), \\ s.t. \quad & 0 \le a^k_i(t) \le A^k_i(t),\ \forall i,k, \end{aligned} \tag{45}$$

where $j^* = argmin\{Q^k_{i,v}(t), \forall v \in \mathcal{D}_k\}$ refers to the shortest feasible queue in *DC* i. Finally, the optimal solution $a^k_i(t)^*$ is

$$a^k_i(t)^* = \begin{cases} A^k_i(t), & Q^k_{i,j^*}(t) \le X^k_i(t)\cdot\delta_k, \\ 0, & \text{otherwise}. \end{cases} \tag{46}$$

The data-transfer and task processing in *LocalAny* are similar to those in *GlobalAny*. However, with *LocalAny*, the tasks can only be processed in the pre-selected destination DCs, rather than any feasible ones.

2) LocalAny_Ext: Similar to *GlobalAny_Ext*, we refer to the algorithm that incorporates the data-transfer acceleration scheme in *LocalAny* as *LocalAny_Ext*. We use it as another benchmark algorithm in the simulations.

3) Benchmark Algorithms: Moreover, we also compare the performance of our proposed algorithms with several existing ones. The first one is to transfer the tasks to their destination DCs directly, *i.e.*, the direct-transfer (*DT*). Specifically, in each TS, the source DC selects the destination DC for each newly-generated task randomly and then network bandwidth is allocated by calculating the multi-source multi-destination maximum flow. If bandwidth is insufficient, the task is buffered in its source DC until there is enough out-bound bandwidth. The second one, *i.e.*, *TEN-SnF*, adopts the store-and-forward scheme with the time-expanded networking technique in [10], and selects each task's destination DC randomly.

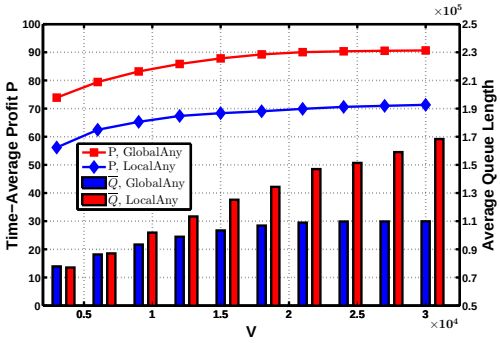

Fig. 5. Impacts of V on *GlobalAny* and *LocalAny*.

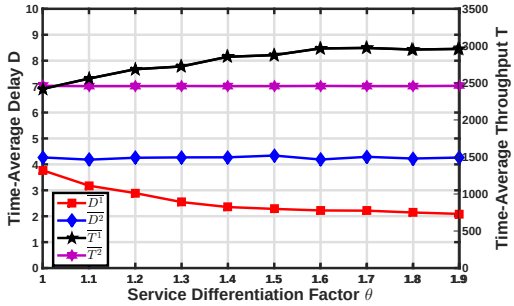

Fig. 6. Impact of task category weight w_1.

B. Parameter Analysis

1) Adjustable Parameter V: We first investigate the impact of parameter V. Fig. 5 shows the time-average profit P for *GlobalAny* and *LocalAny*. We can see that for *GlobalAny*, P increases with V faster when V is smaller. Eventually, when V keeps increasing, P would converge to its optimal value smoothly, which verifies that the time-average profit P can approach to the optimal value arbitrarily within a constant gap $O(\frac{1}{V})$. We notice that *GlobalAny* achieves a higher profit P than *LocalAny*. In Fig. 5, we also plot the average value of the total time-average queue length in each DC, *i.e.*, $\overline{Q} = \frac{1}{|\mathcal{D}|}\sum_{i,k}\overline{Q^k_i(t)}$ for *GlobalAny* and $\overline{Q} = \frac{1}{D}\sum_{i,j,k}\overline{Q^k_{i,j}(t)}$ for *LocalAny*. It can be seen that for the same V, the $\overline{Q}$ from *GlobalAny* is less than that from *LocalAny*, and this advantage becomes more obvious when V increases.

2) Category Weight w_k: As explained in Section IV, *GlobalAny* can provide differentiated services to different task categories by adjusting their weights $\{w_k\}$. To demonstrate this feature, we multiply the weight of the first task category, *i.e.*, w_1, with a parameter θ, and keep the remaining weights unchanged. Here, we use D_1 to denote the average task processing delay of the first category, while the average delay of other categories are denoted as D_{-1}. T_1 and T_{-1} are used to denote the throughput of the first category and other categories, respectively. Fig. 6 shows the results on task processing delay and throughput for different θ, which indicate that D_1 decreases with θ, and T_1 increases with θ. This is because more tasks in the first category can be accepted with a larger w_1, and since we would allocate more bandwidth and IT resources to

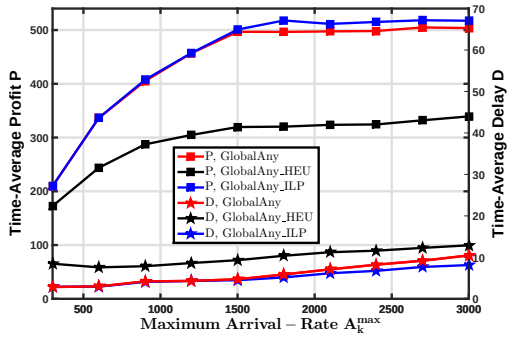

Fig. 7. *GlobalAny* with different bandwidth allocation policies.

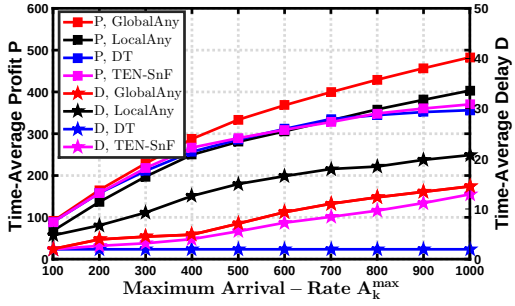

Fig. 8. Comparison with benchmark algorithms in EC2 topology.

this task category, the task processing latency becomes shorter. Meanwhile, D_{-1} and T_{-1} stay almost unchanged for different θ. Hence, we verify that the performance of a task category can be adjusted adaptively.

C. Bandwidth Allocation Policies

We then simulate *GlobalAny* with different bandwidth allocation and data-transfer polices. Here, we denote the algorithm that solves the integer linear programming (ILP) model in Eq. (37) for bandwidth allocation and data-transfer as *GlobalAny_ILP*[1]. According to the discussion in Section IV-B3, we can design a simple data-transfer policy that lets the queue transfer data to the adjacent DCs randomly, if it has the maximum positive $B_{i,v}^k(t)$ on multiple links. The algorithm with this policy is denoted as *GlobalAny_HEU*. We set $V = 3 \times 10^4$ and perform simulations with different maximum arrival rate A_k^{max}. Fig. 7 shows the results on P and D, which indicate that P increases with A_{max} for all the policies. Note that, compared with *GlobalAny_ILP*, *GlobalAny* provides similar results on P and D, while the performance of *GlobalAny_HEU* on them is much worse. Meanwhile, the average computation time to get the bandwidth allocation scheme for a DC in a TS is 32.7 msec, 19.1 msec and 125.3 msec, with *GlobalAny*, *GlobalAny_HEU* and *GlobalAny_ILP*, respectively. Therefore, *GlobalAny* can achieve the similar task scheduling performance as *GlobalAny_ILP*, with much shorter computation time. Even though *GlobalAny_HEU* consumes the shortest computation time, its performance is significantly worse than *GlobalAny* and *GlobalAny_ILP*. To this end, we can see that *GlobalAny* can obtain near-optimal scheduling performance with reasonably short computation time.

[1]CPLEX is used to solve the ILP, and all the simulations in this paper run on a computer with an Intel CPU (I3-2120, 3.30GHz) and 8 GB memory.

D. Comparison with Benchmark Algorithms

Next, the simulations use different maximum arrival rate A_k^{max} to emulate the change of traffic loads. And V is chosen to balance the profit and delay as much as possible for *GlobalAny* and *LocalAny*. Fig. 8 shows the results on the time-average profit and average delay from different algorithms. Apparently, *GlobalAny* achieves the highest profit P among the algorithms, especially when the traffic load is relatively high. The average task processing delay D from *GlobalAny* is shorter than that from *LocalAny*, while it is longer than those from *DT* and *TEN-SnF*. Actually, the results on average task processing delay from *GlobalAny* and *TEN-SnF* are comparable, but *DT* provides fixed and the shortest task processing latency. However, as *DT* drops the requests when the system can not transfer or process them directly and its profit is less than that of *GlobalAny* significantly.

Table III shows the results on the average throughput T and computation time. We can see that T increases with A_k^{max} and *GlobalAny* provides the highest T among the algorithms. As for the computation time, *GlobalAny* is at least one-magnitude faster than *DT* and *TEN-SnF*, which verifies that *GlobalAny* can successfully address the scalability issue of *DT* and *TEN-SnF* and operate in an online manner. Moreover, the computation time of *GlobalAny* keeps steady when A_k^{max} increases, while that of *TEN-SnF* increases with A_k^{max}. This is because *TEN-SnF* invokes more frequent network-expanding operations when A_k^{max} is larger, which is time-consuming.

TABLE III
PERFORMANCE COMPARISONS USING EC2 TOPOLOGY

	Average Throughput ($\times 10^5$)			Computation Time (msec)		
A_k^{max}	300	600	900	300	600	900
GlobalAny	1.46	2.32	2.76	29.5	31.2	30.6
LocalAny	1.27	2.14	2.65	57.2	58.4	59.6
DT	1.40	2.11	2.58	357.6	344.0	327.8
TEN-SnF	1.43	2.17	2.63	734.4	1432.8	1925.0

E. Simulations with B4 Topology

Finally, we perform simulations with the B4 topology to evaluate the proposed algorithms further. We set $A_k^{max} = 100$ and change V to investigate the impacts of parameter V. Fig. 9 shows the time-average profit P and average total queue length $\overline{Q}$ from *GlobalAny*, *GlobalAny_ILP*, and *GlobalAny_Ext*. We observe that both P and $\overline{Q}$ increase with V and then converge to fixed values, which show similar trends as those obtained with the EC2 topology. And the performance of *GlobalAny* on P and $\overline{Q}$ is close to that of *GlobalAny_ILP*. It is interesting to see that the profit from *GlobalAny_Ext* is higher than that from *GlobalAny* and the $\overline{Q}$ from *GlobalAny_Ext* is less than

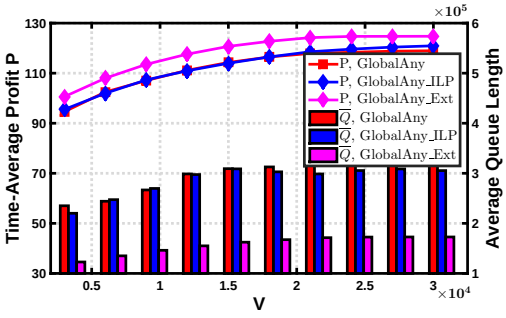

Fig. 9. Impacts of V in B4 topology.

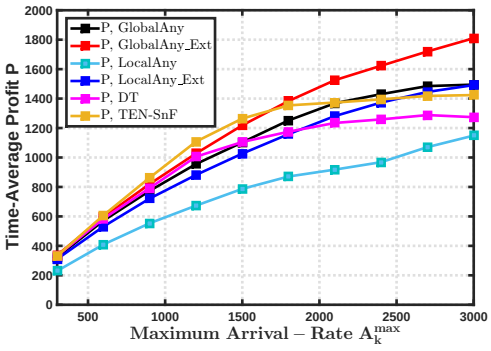

Fig. 10. Time-average profits in B4 topology.

that from *GlobalAny*. The results verify the effectiveness of the data-transfer acceleration scheme.

We also simulate the algorithms with different A_k^{max}. Figs. 10 and 11 show the results on the time-average profit P and average task processing delay D, respectively. It can be seen clearly that the algorithms with the data-transfer acceleration scheme always provide higher profit and shorter delay, when compared with their counterparts without the scheme, especially for *GlobalAny_Ext*. Table IV shows the results on the average throughput T and computation time. We observe that *GlobalAny* and *GlobalAny_Ext* achieve higher throughput than the other algorithms. When comparing it with *GlobalAny*, we find that *GlobalAny_Ext* achieves much higher throughput with just slightly longer computation time. Note that, although the running time does become longer for *GlobalAny_Ext*, it is still acceptable for realizing online operations.

VII. Conclusions

This paper investigated the distributed online scheduling for data-oriented tasks in cloud DC systems. By considering the store-and-forward and anycast schemes, we formulated an optimization problem to maximize the time-average profit and then leveraged the Lyapunov optimization techniques to propose an efficient scheduling algorithm, *i.e.*, *GlobalAny*. We also designed a data-transfer acceleration scheme to reduce the data-transfer latency. Numerical simulations verified that our algorithms can maximize the time-average profit in a distributed online manner. The results also indicated that *GlobalAny* and *GlobalAny_Ext* (*i.e.*, *GlobalAny* with data-transfer acceleration) outperformed several existing algorithms in terms of both time-average profit and computation time.

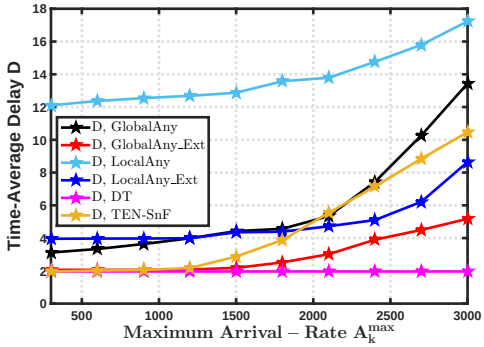

Fig. 11. Time-average task processing delay in B4 topology.

TABLE IV
Performance Comparisons using B4 Topology

	Average Throughput ($\times 10^6$)			Computation Time (msec)		
A_k^{max}	300	600	900	300	600	900
GlobalAny	0.61	1.01	1.13	53.1	56.4	57.1
LocalAny	0.47	0.73	0.89	98.2	98.7	99.6
GlobalAny_Ext	0.65	1.15	1.43	254.4	261.2	263.8
LocalAny_Ext	0.54	0.89	1.09	435.1	457.7	431.6
DT	0.68	1.07	1.19	1962.3	1962.5	1960.5
TEN-SnF	0.73	1.25	1.36	2055.1	3885.1	8847.1

Acknowledgments

This work was supported in part by the NSFC Project 61701472, CAS Key Project (QYZDY-SSW-JSC003), NGBWMCN Key Project (2017ZX03001019-004), China Postdoctoral Science Foundation (2016M602031), and Fundamental Research Funds for the Central Universities (WK2100060021).

References

[1] J. Yao, P. Lu, L. Gong, and Z. Zhu, "On fast and coordinated data backup in geo-distributed optical inter-datacenter networks," *J. Lightw. Technol.*, vol. 33, pp. 3005–3015, Jul. 2015.

[2] P. Lu *et al.*, "Highly-efficient data migration and backup for big data applications in elastic optical inter-datacenter networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sep./Oct. 2015.

[3] P. Lu and Z. Zhu, "Data-oriented task scheduling in fixed- and flexible-grid multilayer inter-DC optical networks: A comparison study," *J. Lightw. Technol.*, vol. 35, pp. 5335–5346, Dec. 2017.

[4] W. Lu, Z. Zhu, and B. Mukherjee, "Optimizing deadline-driven bulk-data transfer to revitalize spectrum fragments in EONs," *J. Opt. Commun. Netw.*, vol. 7, pp. B173–B183, Dec. 2015.

[5] X. Yi, F. Liu, J. Liu, and H. Jin, "Building a network highway for big data: architecture and challenges," *IEEE Netw.*, vol. 28, pp. 5–13, Jul. 2014.

[6] W. Lu and Z. Zhu, "Malleable reservation based bulk-data transfer to recycle spectrum fragments in elastic optical networks," *J. Lightw. Technol.*, vol. 33, pp. 2078–2086, May 2015.

[7] W. Lu, Z. Zhu, and B. Mukherjee, "On hybrid IR and AR service provisioning in elastic optical networks," *J. Lightw. Technol.*, vol. 33, pp. 4659–4669, Nov. 2015.

[8] A. Castro *et al.*, "Experimental demonstration of heterogeneous cross stratum broker for scientific applications," in *Proc. of OFC 2016*, pp. 1–3, Mar. 2016.

[9] X. Xie *et al.*, "Evacuate before too late: Distributed backup in inter-DC networks with progressive disasters," *IEEE Trans. Parallel Distrib. Syst., in Press*, 2017.

[10] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with Netstitcher," in *Proc. of SIGCOMM 2011*, pp. 74–85, Aug. 2011.

[11] Z. Zhu *et al.*, "Jitter and amplitude noise accumulations in cascaded all-optical regenerators," *J. Lightw. Technol.*, vol. 26, pp. 1640–1652, Jun. 2008.

[12] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," in *Proc. of SIGCOMM 2013*, pp. 3–14, Aug. 2013.

[13] S. Li, W. Lu, X. Liu, and Z. Zhu, "Fragmentation-aware service provisioning for advance reservation multicast in SD-EONs," *Opt. Express*, vol. 23, pp. 25 804–25 813, Oct. 2015.

[14] M. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan and Claypool Publishers, 2010.

[15] H. Goudarzi and M. Pedram, "Force-directed geographical load balancing and scheduling for batch jobs in distributed datacenters," in *Proc. of CLUSTER 2013*, pp. 1–8, Sep. 2013.

[16] P. Lu, Q. Sun, K. Wu, and Z. Zhu, "Distributed online hybrid cloud management for profit-driven multimedia cloud computing," *IEEE Trans. Multimedia*, vol. 17, pp. 1297–1308, Aug. 2015.

[17] W. Lu and Z. Zhu, "Dynamic service provisioning of advance reservation requests in elastic optical networks," *J. Lightw. Technol.*, vol. 31, pp. 1621–1627, May 2013.

[18] P. Lu, Q. Ling, and Z. Zhu, "Maximizing utility of time-constrained emergency backup in inter-datacenter networks," *IEEE Commun. Lett.*, vol. 20, pp. 890–893, May 2016.

[19] K. Wu, P. Lu, and Z. Zhu, "Distributed online scheduling and routing of multicast-oriented tasks for profit-driven cloud computing," *IEEE Commun. Lett.*, vol. 20, pp. 684–687, Apr. 2016.

[20] Y. Wang, S. Su, A. Liu, and Z. Zhang, "Multiple bulk data transfers scheduling among datacenters," *Comput. Netw.*, vol. 68, pp. 123 – 137, Aug. 2014.

[21] M. Chowdhury, M. Rahman, and R. Boutaba, "ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Trans. Netw.*, vol. 20, pp. 206–219, Jan. 2012.

[22] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.

[23] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proc. of INFOCOM 2014*, pp. 1–9, Apr. 2014.

[24] H. Jiang, Y. Wang, L. Gong, and Z. Zhu, "Availability-aware survivable virtual network embedding (A-SVNE) in optical datacenter networks," *J. Opt. Commun. Netw.*, vol. 7, pp. 1160–1171, Dec. 2015.

[25] W. Fang *et al.*, "Joint spectrum and IT resource allocation for efficient vNF service chaining in inter-datacenter elastic optical networks," *IEEE Commun. Lett.*, vol. 20, pp. 1539–1542, Aug. 2016.

[26] M. Zeng, W. Fang, and Z. Zhu, "Orchestrating tree-type VNF forwarding graphs in inter-DC elastic optical networks," *J. Lightw. Technol.*, vol. 34, pp. 3330–3341, Jul. 2016.

[27] Y. Wang, P. Lu, W. Lu, and Z. Zhu, "Cost-efficient virtual network function graph (vNFG) provisioning in multidomain elastic optical networks," *J. Lightw. Technol.*, vol. 35, pp. 2712–2723, Jul. 2017.

[28] J. Liu *et al.*, "On dynamic service function chain deployment and readjustment," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, pp. 543–553, Sept. 2017.

[29] R. Kettimuthu, G. Vardoyan, G. Agrawal, and P. Sadayappan, "Modeling and optimizing large-scale wide-area data transfers," in *Proc. of CCGrid 2014*, pp. 196–205, May 2014.

[30] C. Hong *et al.*, "Achieving high utilization with software-driven WAN," in *Proc. of SIGCOMM 2013*, pp. 15–26, Aug. 2013.

[31] Y. Wu *et al.*, "Orchestrating bulk data transfers across geo-distributed datacenters," *IEEE Trans. Cloud Comput., in Press*, pp. 1–14, 2015.

[32] S. Li *et al.*, "Towards operational cost minimization in hybrid clouds for dynamic resource provisioning with delay-aware optimization," *IEEE Trans. Serv. Comput.*, vol. 8, pp. 398–409, May 2015.

[33] R. Urgaonkar, B. Urgaonkar, M. Neely, and A. Sivasubramaniam, "Optimal power cost management using stored energy in data centers," in *Proc. of SIGMETRICS 2011*, pp. 221–232, Jun. 2011.

[34] F. Liu *et al.*, "On arbitrating the power-performance tradeoff in SaaS clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, pp. 2648–2658, Oct. 2014.

[35] L. Georgiadis, M. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, pp. 1–144, Apr. 2006.

[36] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *IEEE/ACM Trans. Netw.*, vol. 19, pp. 841–854, Jun. 2011.

[37] L. Bui, R. Srikant, and A. Stolyar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in *Proc. of INFOCOM 2009*, pp. 2936–2940, Apr. 2009.

[38] R. Urgaonkar, U. Kozat, K. Igarashi, and M. Neely, "Dynamic resource allocation and power management in virtualized data centers," in *Proc. of NOMS 2010*, pp. 479–486, Apr. 2010.

[39] D. Pisinger, "Algorithms for knapsack problems," Ph.D. dissertation, University of Copenhagen, Feb. 1995.

[40] Y. Feng, B. Li, and B. Li, "Jetway: Minimizing costs on inter-datacenter video traffic," in *Proc. of ACM MM 2012*, pp. 259–268, Oct. 2012.