Traffic-Aware Virtual Machine Migration in Topology-Adaptive DCN

Yong Cui, Zhenjie Yang, Shihan Xiao, Xin Wang, and Shenghui Yan

***Abstract*— Virtual machine (VM) migration is a key technique for network resource optimization in modern data center networks. Previous work generally focuses on how to place the VMs efficiently in a static network topology by migrating the VMs with large traffic demands to close servers. As the flow demands between VMs change, however, a great cost will be paid for the VM migration. In this paper, we propose a new paradigm for VM migration by dynamically constructing adaptive topologies based on the VM demands to lower the cost of both VM migration and communication. We formulate the traffic-aware VM migration problem in an adaptive topology and show its NP-hardness. For periodic traffic, we develop a novel progressive-decompose-rounding algorithm to schedule VM migration in polynomial time with a proved approximation ratio. For highly dynamic flows, we design an online decision-maker (ODM) algorithm with proved performance bound. Extensive trace-based simulations show that PDR and ODM can achieve about four times flow throughput among VMs with less than a quarter of the migration cost compared to other state-of-art VM migration solutions. We finally implement an OpenvSwitch-based testbed and demonstrate the efficiency of our solutions.**

***Index Terms*— Data center network, VM migration, wireless communication.**

I. Introduction

With the proliferation of cloud computing, virtualization has become a popular practice in the design of data centers. Without considering the specific running status of the user applications, network operators can simply migrate the VMs to achieve better resource utilization, failure tolerance, load balancing, energy efficiency, and so on [1], [2].

As VM migration is a key feature for elastic computing and plays a critical role in data center operation, intensive recent efforts have been made to minimize the cost of migrating one or several VMs from an initial placement to the target one in the **migration phase** [3]–[5]. In order to benefit from high-performance live migration, there is also a need to reduce the communication cost among VMs under the optimized target placement, which we refer as **communication phase**.

Manuscript received December 24, 2016; revised June 24, 2017; accepted August 4, 2017; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Wierman. Date of publication September 8, 2017; date of current version December 15, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61422206 and in part by the Tsinghua National Laboratory for Information Science and Technology. The work of X. Wang was supported by the NSF CNS under Grant 1526843. *(Corresponding author: Yong Cui.)*

Y. Cui, Z. Yang, S. Xiao, and S. Yan are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: cuiyong@tsinghua.edu.cn; yangzj15@mails.tsinghua.edu.cn; xiaosh12@mails.tsinghua.edu.cn; yansh14@mails.tsinghua.edu.cn).

X. Wang is with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11790 USA (e-mail: x.wang@stonybrook.edu).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Digital Object Identifier 10.1109/TNET.2017.2744643

With the increasing trend of running communication-intensive applications in data centers [6], [7], intuitively, the VMs with large traffic demands should be migrated to servers in close proximity (in the topological sense). The observation of the VM traffic stability at large timescales in [6] confirms the feasibility of lowering the communication cost by optimizing the VM placement based on traffic statistics, i.e., the so-called *traffic-aware* VM placement. Recent efforts in [8] and [9] further take the routing options into account and propose to jointly optimize the VM placement and proper routing paths for the VM communication, which achieves a consistent and significant improvement over the common practice in data centers.

While great efforts have been made on minimizing the cost of the migration phase or the communication phase, few consider their joint optimization. However, there may be a trade-off to reduce these two costs. For example, a *good* placement obtained by minimizing the communication cost may be a *poor* option for the migration, due to the high cost of reshuffling existing VMs [10]. To address this challenge, our basic motivation is that, if the network topology can be changed dynamically with reconfigurable links, the servers which hold the VMs with large traffic demands can be bridged together with direct short links. In this way, little migration is needed while the communication cost is reduced remarkably.

In this paper, we explore a novel paradigm, called the *topology-adaptive DCN*, to lower the cost of both the migration phase and communication phase. With the advance of software-defined networking (SDN) technologies in DCN, recent studies have shown a great potential to implement a reconfigurable network topology [11]–[19]. Nowadays, there are mainly two technologies to achieve this topology-adaptive objective. The first is adding the 60GHz wireless radios or the Free-Space Optics (FSO) [11], [12], [14], [16], [17]. The second is adding the optical circuit switches (OCS) with fast circuit switching to adapt the topology [13], [15], [18], [19]. The core of the above technologies is to collect traffic information through the OpenFlow protocol on the SDN platform and then build *configurable links* on-demand by the SDN controller [14]. As topology reconfiguration can be achieved within a few microseconds, it would have little impact on the total performance gains [11], [16], [17].

To the best of our knowledge, this is the first work to study VM migration with a reconfigurable network topology. Specifically, to minimize the total cost in the VM migration and communication, we will jointly make three challenging decisions for optimal performance in this work: (1) the migration decision, i.e., which VMs should be migrated to which physical servers, with respect to the server capacity; (2) the topology decision, i.e., which configurable links should be built to implement a suitable topology for the current VM demands, taking into account the link conflict constraints; (3) the routing decision, i.e., how should the VMs route their

traffic demands over the newly configured network topology, with respect to the link capacity.

Existing work on the joint optimization of the VM placement with other metrics like routing, link utilization or energy consumption [6], [8], [9], [20] are generally shown to be NP-hard and only design heuristic algorithms. In this paper, we will show how to address this general challenge with a proved approximation ratio using an abstracted modeling solution. The main contributions of our work are as follows:

- We concurrently consider the costs of migration phase and communication phase, and formulate the traffic-aware VM migration problem in an adaptive topology, which is shown to be NP-hard.
- For data centers with periodic traffic, we propose a novel progressive-decomposition-rounding (PDR) algorithm to solve the migration problem and prove its approximation ratio in three steps. We show that our technique can be extended to solve the VM migration optimization under different topology adaption technologies in DCNs with proved approximation ratios.
- For data centers with random and dynamic flows, we design an online decision-maker (ODM), which applies the Lyapunov optimization framework to schedule VM migration requests in real-time. We further analyze and prove the performance bound of ODM.
- We conduct real-trace based evaluations to demonstrate the efficiency of our solutions under various scenarios, and then validate the feasibility of our solutions in improving the flow performance with implementation over an OpenvSwitch-based testbed.

The remainder of this paper is organized as follows. Section II illustrates our problem formulation. In Section III, we provide scheduling analysis and present an offline algorithm, PDR. We further design an online decision-maker (ODM) to schedule VM migration requests in real-time in Section IV. We evaluate the performance of our scheduling algorithms in Section V and make deployment discussions in Section VI. Finally, we introduce the related work in Section VII, and conclude our work in Section VIII.

II. Problem Formulation

In this section, we introduce the motivation of our work as well as the basic system model, and then formulate the migration problem.

A. Motivating Example

To begin with, we introduce an example to illustrate how we can address the performance bottleneck of VM migration using the topology adaption. In Fig. 1, we compare the migration and communication costs for different migration solutions. There are two VMs, A and B with a memory size of 100 MBytes each, while the communication traffic between them is 400 MBytes. Suppose the servers are very busy that each server can only accommodate at most one VM. For simplicity, one unit of network cost is defined as routing one MByte of the traffic over one hop. At the same time, the migration paths are pre-specified as the ones with the least number of hops, and wireless links are configured to serve in the communication phase.

In Fig. 1, if VM A and B communicate without any migration, the migration cost is zero and the communication cost is 400MB $\times$ 6 hops=2400, then the total cost is 2400.

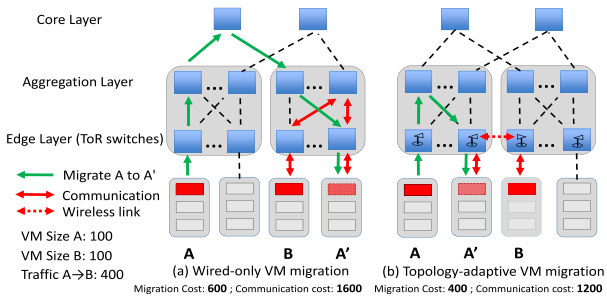


Fig. 1. Cost comparison of different VM migration solutions. (a) Wired-only VM migration. (b) Topology-adaptive VM migration.

We call this an *original cost* in the following. With an optimal migration performed over a static wired topology as shown in Fig. 1a, we migrate A to A$'$ (green solid line) and A$'$ communicates with B (red solid line). Then a lower communication cost 1600 is achieved at the cost of a higher migration cost at 600, which reduces the total cost by only 8%. However, if we take a topology-adaptive strategy as in Fig. 1b, we migrate A to A$'$ (green solid line) at a migration cost of 400 and A$'$ communicates with B (red solid and dotted line) at a communication cost of 1200, the total cost is reduced by 33% compared with the original cost. Finally, if wireless has a larger coverage, the optimal topology-adaptive solution in Fig. 1 is to build a direct wireless link between the ToR (Top-of-Rack) switches of A and B, which produces the lowest communication cost of 1200 and zero migration cost. In this case, we reduce the total cost by 50%. We can see that a single link adaptation can achieve a high performance benefit for VM migration and communication. We note that a more flexible strategy which mixes multiple wireless links and wired links to construct hybrid paths for VM migration and communication is possible to gain higher benefits. We will formulate the flexible optimization problem formally in the following and show its non-triviality.

B. VM Placement

Motivated by the above example, the objective of our system is to reduce the total cost of VM migration and communication. We take the VM demands in one period as an input matrix $\mathcal{D}$ for our system. Each element in the matrix denotes a *VM-level* flow $f_{pq} \in \mathcal{F}$ from the VM M_p to M_q with a traffic demand d_{pq} ($\mathcal{F}$ is the flow set).

At a high level, each period consists of two phases: (1) the *migration phase* to transform the initial VM placement $\mathcal{A}$ to a new placement $\mathcal{B}$; (2) the *communication phase* that the VMs communicate with each other under the new placement $\mathcal{B}$. At the beginning of each period, our system uses an input demand matrix $\mathcal{D}$ to compute a new VM placement $\mathcal{B}$, a new network topology and also the routing plan. The placement $\mathcal{B}$ is used as the migration objective in the migration phase, while the new topology and routing plan are deployed at the head of communication phase to facilitate the VM communications.

We assume existing capacity tools (CPU/memory based) has determined the number of VMs that a server can host [6]. Hence we use one CPU/memory allocation on a server as the basic unit for VM placement and denote it as a *slot*. We denote the active VMs in one period as a VM set $\mathcal{M} = \{M_1, M_2, ..., M_n\}$ and the available slot set as $\mathcal{L} = \{L_1, L_2, ..., L_m\}$, $n \leq m$. Then a *VM placement* can be

denoted as a one-to-one mapping from a VM set to a slot set. Without loss of generality, we index the slots by their VM index in the initial placement $\mathcal{A}$, i.e., the VM M_i is located at slot L_i under the placement $\mathcal{A}$. For the new placement $\mathcal{B}$, we use a binary variable $y_{ij} \in \{0,1\}$ to denote whether VM M_i is placed in the slot L_j under the placement $\mathcal{B}$.

C. Adaptive Topology

We denote the topology of DCN as a graph $G(V,E)$, where V denotes all the switches and servers, and E denotes all the links connecting them including the configurable wireless links. Let S denote all the servers, then the set $V \backslash S$ denotes all the switches. The main challenge of routing in an adaptive topology is to describe the flow conservation constraint when the new VM placement $\mathcal{B}$ is not decided yet. In the following, we present a novel linear formulation to address this issue.

Let a binary variable x_{ij}^{pq} denote whether a flow f_{pq} routes through the link e_{ij}, and b_{ij} is the link capacity of link $e_{ij} \in E$. The flow conservation constraints under the new VM placement $\mathcal{B} = \{y_{ij}\}$ can be equally translated to the following linear equations:

$$\forall v_i \in S, f_{pq} \in \mathcal{F}:$$
$$\sum_{e_{ij} \in E} x_{ij}^{pq} - \sum_{e_{ji} \in E} x_{ji}^{pq} = \sum_{k \in \widehat{L}(i)} y_{pk} - \sum_{k \in \widehat{L}(i)} y_{qk} \quad (1)$$
$$\forall v_i \in V \setminus S, f_{pq} \in \mathcal{F}:$$
$$\sum_{e_{ij} \in E} x_{ij}^{pq} - \sum_{e_{ji} \in E} x_{ji}^{pq} = 0 \quad (2)$$

where $\widehat{L}(i)$ denotes all the slots in server i.

The constraint of the equation (1) applies when a node i in the server set S is the source or destination sever. The right-hand side (RHS) of the equation equals 1 or -1, so that the traffic out of the source server and the traffic into the destination server equals the flow demand. Otherwise, if node i in S is not the source or destination server, the RHS equals 0. Similarly, when the node i is an intermediate switch, the flows are constrained by the equation (2) according to the flow conservation rule.

D. Conflict Constraints for Configurable Links

Although building more configurable links would allow for more flexibility to find a better solution, in a topology-adaptive DCN, configurable links are not created without constraints. For example, there exists interference among the wireless links, while for the OCSs and FSOs, there are conflicts for creating links at the same port because each port can only support at most one link. In the following, we take the *configurable links* as the wireless links and discuss issues to consider for other mechanisms in Section VI.

Without loss of generality, we construct a *conflict graph* $G_c(V_c, E_c)$ to describe the conflict relations among all the configurable wireless links. In the following, we simplify the analysis by assuming the binary interference model [21], and we will show how to extend our solution to the SINR interference model in Section VI. In the binary interference model, two configurable links e_1 and e_2 are said to have conflicts if either e_1 is in the interference range of e_2 or vice versa. Let each configurable link e be a vertex in G_c, and add an edge (e_1, e_2) between any two vertices e_1 and e_2 if and only if they conflict with each other. Finally, we also add all the fixed wired links into the conflict graph, i.e., each wired link is a vertex in G_c but has no edges with other vertices.

TABLE I
NOTATIONS AND DEFINITIONS

Notations	Definitions
$\mathcal{M}$	the VM set $\{M_1, M_2, ..., M_n\}$
$\mathcal{L}$	the slot set $\{L_1, L_2, ..., L_m\}$
x_{ij}^{pq}	{0,1}: whether flow f_{pq} routes through link e_{ij}
y_{ij}	{0,1}: whether VM M_i is placed in slot L_j
z_{ij}	{0,1}: whether link e_{ij} is selected to build
f_{pq}	A flow from source VM M_p to destination VM M_q
d_{pq}	Flow demand of flow f_{pq}
b_{ij}	Link capacity of link e_{ij}

In the conflict graph, two vertices are said to be *independent* if there is no direct edge connecting them. An *independent set* (IS) in a conflict graph $G_c(V_c, E_c)$ is defined as a vertex subset of V_c where any two vertices are *independent*. Hence a feasible solution for setting up configurable links in G is to select an IS in the conflict graph G_c, i.e., we have

$$z_{ij} \in \{0,1\}, \quad \forall e_{ij} \in E \quad (3)$$
$$z_{ij} + z_{uv} \leq 1, \quad \forall (e_{ij}, e_{uv}) \in E_c \quad (4)$$

where z_{ij} denotes whether the link e_{ij} is selected to build.[1] and constraint (4) ensures the independence among links. For clarity, we list all the related notations and definitions in Table I.

In the following, we give the relationship between the flow variable x_{ij}^{pq} and link variable z_{ij}. Since a link is considered to be built only if there exists flow routing over it, we have the *link existence constraint*: $x_{ij}^{pq} \leq z_{ij}, \forall f_{pq} \in \mathcal{F}, e_{ij} \in E$. Further, we have the *link capacity constraint* for routing: $\sum_{f_{pq} \in \mathcal{F}} d_{pq} x_{ij}^{pq} \leq b_{ij}, \forall e_{ij} \in E$, which means the total traffic demand of all the flows which route over the link e_{ij} is within the link capacity b_{ij}. Benefited from the binary property of $x_{ij}^{pq} \in \{0,1\}$ and $z_{ij} \in \{0,1\}$, the above two constraints can be combined and equally translated in a simplified way:

$$\sum_{f_{pq} \in \mathcal{F}} d_{pq} x_{ij}^{pq} / b_{ij} \leq z_{ij}, \quad \forall e_{ij} \in E \quad (5)$$

E. Migration and Communication Cost

1) Migration Cost: To evaluate the cost of migration phase, we use a_{ij} to denote the general migration cost by migrating the VM M_i from its previous slot to a new slot L_j. When applying our model into practice, the network operators are free to use specific cost metrics they prefer in the migration phase to set a_{ij}, such as the total migration time, the service downtime or the migration traffic size, etc. Generally, the migration cost a_{ij} can be easily obtained based on the measured metrics in data centers by the SDN platform, e.g., the migration time and service downtime can be estimated by the VM memory sizes, the allocated bandwidth and the measured dirty page rates of the VMs [5]. Moreover, the VM migration paths are all pre-specified based on the current network topology, and the value of a_{ij} can be set as the product of memory size and migration path length.

Therefore, the total network cost of the migration phase from the placement $\mathcal{A}$ to a new placement $\mathcal{B}$ (named *migration*

[1]For a wired/wireless link, $z_{ij} = 0$ means no flow is allowed to route over this link; and $z_{ij} = 1$ means flows can route over this link.

cost) can be computed as

$$\mathcal{C}_{\mathcal{AB}} = \sum_{M_i \in \mathcal{M}} \sum_{L_j \in \mathcal{L}} a_{ij} y_{ij} \tag{6}$$

2) Communication Cost: To evaluate the communication cost among VMs, the *communication distance* between server i and j can be modeled with the routing hops between them [6]. Then the cost of VM communication can be described by the product of the traffic demand and communication distance.

For a flow $f_{pq} \in \mathcal{F}$, the communication distance is then calculated as $\sum_{e_{ij} \in E} x_{ij}^{pq}$. Hence we have the total cost of VM communications (named *communication cost*):

$$\mathcal{T}_{\mathcal{B}} = \sum_{f_{pq} \in \mathcal{F}} \{ d_{pq} \sum_{e_{ij} \in E} x_{ij}^{pq} \} = \sum_{f_{pq} \in \mathcal{F}} \sum_{e_{ij} \in E} d_{pq} x_{ij}^{pq} \tag{7}$$

F. Problem Formulation

The objective of our system is to minimize the sum of the migration cost and communication cost during each scheduling period. We use a weight parameter β to show the trade-off between these two costs, which can be adjusted according to the network operators' preference on the two costs. Specially, the model still applies if only one cost is considered by setting β to zero or a large number. The controller input is n VMs and each VM-level flow $f_{pq} \in \mathcal{F}$ is attached with a flow demand d_{pq}. Therefore, we generate the following joint optimization problem $\mathcal{P}_0$:

$$\begin{aligned}
&\min \ \mathcal{C}_{\mathcal{AB}} + \beta \times \mathcal{T}_{\mathcal{B}} \\
&\text{subject to} \sum_{v_j \in \mathcal{L}} y_{ij} = 1, \quad \forall M_i \in \mathcal{M} && (8) \\
&\qquad \sum_{M_i \in \mathcal{M}} y_{ij} \leq 1, \quad \forall L_j \in \mathcal{L} && (9) \\
&\qquad x_{ij}^{pq} \in \{0,1\}, \quad y_{ij} \in \{0,1\} && (10) \\
&\qquad \text{Constraints} \ \ (1),(2),(3),(4),(5) && (11)
\end{aligned}$$

where constraint (8)(9) are the VM placement constraints that each VM must be hosted by only one slot while one slot can host at most one VM. Constraint (1)(2) are the flow conservation constraints. Constraint (3)(4)(5) are appended to take into account the conflict constraints among configurable links and link capacity. In the following, we use *total cost* to denote the optimization objective, i.e., the sum of the migration cost ($\mathcal{C}_{\mathcal{AB}}$) and the weighted communication cost ($\beta \times \mathcal{T}_{\mathcal{B}}$).

By solving this optimization problem, the controller outputs three sets of decisions: (1) the new VM placement $\mathcal{B} = \{y_{ij}\}$; (2) the configurable links $\{z_{ij}\}$ selected to build; (3) the routing paths selected for VM-level flows $\{x_{ij}^{pq}\}$ under the placement $\mathcal{B}$. Since the general binary integer problem is NP-hard [22], there are three sets of 0-1 integer variables in the problem $\mathcal{P}_0$ to construct its difficulty. The first is the unsplittable property of VMs indicated by $y_{ij} \in \{0,1\}$, i.e., one VM can be migrated to only one slot. The second is the unsplittable property of configurable links indicated by $z_{ij} \in \{0,1\}$, i.e., only one IS is selected to build. The third is the unsplittable flow property indicated by $x_{ij}^{pq} \in \{0,1\}$, which is required by the high performance of flows in DCNs [23].

By constructing an instance of problem $\mathcal{P}_0$ with no wireless and VM migration, we generate the following theorem:

Theorem 1 (NP-Hardness): The joint optimization problem $\mathcal{P}_0$ is NP-hard.

Proof: See the detailed proof in Appendix A. ■

III. Offline Scheduling Design

A. Design Overview

Since the original problem $\mathcal{P}_0$ is NP-hard, the natural question is: *can we develop a polynomial-time algorithm to solve it with proved approximation ratio?* The key challenge comes from the binary nature of the three sets of variables $(x_{ij}^{pq}, y_{ij}, z_{ij})$, which are coupled with each other. Specifically, the flow conservation constraint (1) closely couples the flow variable x_{ij}^{pq} to the placement variable y_{ij} while the link capacity constraint (5) couples the flow variable x_{ij}^{pq} to the link variable z_{ij}. Conventional techniques using *relax-and-rounding* have the potential to solve the binary integer problems (BIP) with a single set of 0-1 variables, however, they are not suitable for the joint BIP as $\mathcal{P}_0$. There are several challenges to ensure the performance guarantee when considering multiple sets of 0-1 variables. First, since the three sets of binary variables have quite different properties and constraints, a simple relaxation that equally relaxes each binary variable to a linear variable in $[0,1]$ can not give the performance guarantee during the relaxation. Second, since the three sets of variables are closely coupled with each other in the constraints, it is difficult to round all of them directly without any conflicts on the constraints, not to mention guaranteeing any approximation ratio.

To address the above challenges, we develop a novel progressive-decomposition-rounding (PDR) algorithm to solve $\mathcal{P}_0$. The overview of PDR is presented in **Algorithm 1**. First, we develop techniques to relax the three sets of 0-1 variables based on their different constraints so that a constant approximation ratio is guaranteed during the relaxation (Section III-B). Next, rather than rounding the LP solution directly as a whole, we propose to decompose and round the three sets of variables one by one with the performance guarantee based on their specific properties (Section III-C). Finally, we address the challenge on combining the above *progressive* approximation results together to achieve the complete approximation to the original problem $\mathcal{P}_0$ (Section III-D). In the following, we will introduce the technical details of each step in the PDR.

B. Relaxation of Integer Variables

In this section, we will show how to relax the 0-1 integer variables in the problem $\mathcal{P}_0$. To begin with, we show the motivation of our technique with a basic problem analysis. In problem $\mathcal{P}_0$, we can see that the variable y_{ij} has its *individual* constraints (8)(9) that define a valid VM placement, and also a *combined* constraint (1) that is coupled with the variable x_{ij}^{pq} to ensure the flow conservation. Hence a simple linear relaxation of variable y_{ij} to $[0,1]$ means that a VM is allowed to be split and migrated to multiple slots. Based on equation (1), this relaxation operation will drive the variable x_{ij}^{pq} to be relaxed in the same way, which means a flow is allowed to be split over multiple routing paths. For x_{ij}^{pq}, the main concern raised to this linear relaxation is that it is coupled with the variable z_{ij} in the constraint (5) to ensure that the flow routes over a valid link within its capacity. When looking into the inequality (5), we can see that the expression of x_{ij}^{pq} is limited by an upper bound as z_{ij}. On the other hand,

Algorithm 1 PDR: **P**rogressive-**D**ecomposition-**R**ounding

1: Relax three sets of 0-1 variables: $\{x_{ij}^{pq}\}, \{y_{ij}\}, \{z_{ij}\}$
2: Solve the relaxed LP problem to get the fractional solution
3: Decompose and round an VM placement
4: Solve the LP with fixed VM placement, then decompose and round an IS
5: Solve the LP with fixed VM placement and fixed IS, then decompose and round the routing paths

z_{ij} is characterized by its *individual* constraints (3)(4) to define a valid wireless link setup. Hence, we can address the concern of relaxation impact of x_{ij}^{pq} in constraint (5) using the solution space described by the constraints (3)(4) to be related to z_{ij}.

Following the above analysis, we first relax the 0-1 variables x_{ij}^{pq} and y_{ij} by the linear constraints $x_{ij}^{pq} \in [0,1], y_{ij} \in [0,1]$. Constraints (3)(4) define the 0-1 variable z_{ij} as an IS. We use the *incidence vector* to represent an IS, i.e., a vector whose j^{th} element is 1 if and only if the vertex v_j is an element of the IS. Let $\mathbf{z}$ denote the stacked vector of $\{z_{ij}\}$. Since each incidence vector represents an integer point in the vector space of $\mathbf{z}$, all the incidence vectors of ISs in G_c denote an integer point set $\overline{P}$. Thus the constraints (3)(4)(5) can be equally translated as $\sum_{f_{pq} \in \mathcal{F}} d_{pq} x_{ij}^{pq}/b_{ij} \leq z_{ij}$ and $\mathbf{z} \in \overline{P}$, i.e., in a simplified way it can be written as $\{\sum_{f_{pq} \in \mathcal{F}} d_{pq} x_{ij}^{pq}/b_{ij}\} \in \overline{H}$, where $\overline{H}$ denotes the polytope defined by constraints (3)(4)(5).

Intuitively, we can relax the point set $\overline{P}$ as an *independence set polytope* P, i.e., the convex hull of all the integer points in $\overline{P}$. Hence we have $\overline{H} \subseteq P$ and the ideal relaxation that $\{\sum_{f_{pq} \in \mathcal{F}} d_{pq} x_{ij}^{pq}/b_{ij}\} \in P$. However, the polytope P is generally not polynomial-representable for an arbitrary conflict graph G_c. In the following, we will show that if we can approximate P with another polynomial-representable polytope S within a constant ratio μ, i.e., $S \subseteq P \subseteq \mu S$, then we are able to obtain an μ-approximation relaxation of the original problem $\mathcal{P}_0$.

Following the above relaxation procedures, the formulation of the relaxed LP problem $\widetilde{\mathcal{P}}_0$ can be abstracted as follows:

$$\begin{aligned} &\min\ O(X,Y) \\ &\text{subject to } X \in S && (12)\\ &\qquad Y \in U && (13)\\ &\qquad W(X,Y) = 0 && (14)\end{aligned}$$

where X denotes the vector of variables $\{x_{ij}^{pq}\}$, and Y denotes the vector of variables $\{y_{ij}\}$. The polytope S denotes the μ-approximation of polytope P that is described by original constraints (3)(4)(5) with respect to x_{ij}^{pq}. The polytope U is defined by the original constraints (8)(9) and $y_{ij} \in [0,1]$ with respect to y_{ij}. The original constraints (1)(2) are abstracted by a linearly-weighted sum of vector X and Y as described by $W(X,Y)$, and the optimization objective is abstracted by another linearly-weighted sum of vector X and Y as $O(X,Y)$.

Theorem 2 (Relaxation Guarantee): The optimal solution of the relaxed LP problem $\widetilde{\mathcal{P}}_0$ is μ-approximation of the optimal solution of the target original problem $\mathcal{P}_0$, i.e., $\widetilde{\lambda}_0 \leq \mu\lambda_0$ ($\mu \geq 1$), where $\widetilde{\lambda}_0$ is the optimal objective value of $\widetilde{\mathcal{P}}_0$ and λ_0 is the optimal objective value of $\mathcal{P}_0$.

Proof: Following the above abstraction principle, we construct another problem $\widetilde{\mathcal{P}}_0'$ that formulated as below: minimizing $O(X,Y)$ and subject to $X \in \mu S$, $Y \in \mu U$ and $W(X,Y) = 0$. The optimal solution of $\widetilde{\mathcal{P}}_0'$ is denoted as (X^*, Y^*) with the objective value $\widetilde{\lambda}_0'$. First, we will show that $(\mu X^*, \mu Y^*)$ is a solution of $\widetilde{\mathcal{P}}_0$. Comparing $\widetilde{\mathcal{P}}_0$ and $\widetilde{\mathcal{P}}_0'$, the polytope of X and Y in $\widetilde{\mathcal{P}}_0'$ is μ times that in $\widetilde{\mathcal{P}}_0$. Since the problem is the minimization problem, $(\mu X^*, \mu Y^*)$ is in the polytope of $\widetilde{\mathcal{P}}_0$ with respect to X and Y. Further, the linear constraint $W(X,Y) = 0$ is still satisfied when the vector X and Y are all scaled with a constant factor μ. Hence $(\mu X^*, \mu Y^*)$ is a solution of $\widetilde{\mathcal{P}}_0$. Finally, since $\widetilde{\mathcal{P}}_0$ and $\widetilde{\mathcal{P}}_0'$ have the same linearly-weighted objective that $O(\mu X^*, \mu Y^*) = \mu O(X^*, Y^*)$, we have $\widetilde{\lambda}_0 = \mu\widetilde{\lambda}_0'$.

Let all the incidence vectors[2] of the variables $\{y_{ij}\}$ denote an integer point set $\overline{U}$, which is defined by the constraints (8)(9) and $y_{ij} \in \{0,1\}$. Because U forms a matching polytope of y_{ij} in the bipartite graph of mapping the VMs to slots, the matching polytope U is exactly the convex hull of all the integer points in $\overline{U}$ [24]. Then the original problem $\mathcal{P}_0$ can be formulated as: minimizing $O(X,Y)$ and subject to $X \in \overline{H}$, $Y \in \overline{U}$ and $W(X,Y) = 0$. Let $Conv(\overline{P})$ denote the convex hull of point set $\overline{P}$, since $\overline{H} \subseteq Conv(\overline{P}) = P \subseteq \mu S$ and $\overline{U} \subseteq Conv(\overline{U}) = U \subseteq \mu U$ ($\mu \geq 1$), we have that any solution of problem $\mathcal{P}_0$ is the solution of problem $\widetilde{\mathcal{P}}_0'$. Then let λ_0 be the objective value of the optimal solution of problem $\mathcal{P}_0$, we have $\widetilde{\lambda}_0' \leq \lambda_0$. Therefore, we have $\widetilde{\lambda}_0 = \mu\widetilde{\lambda}_0' \leq \mu\lambda_0$. This completes the proof. ■

There is a group of research efforts on how to construct the approximation polytope S [21], [25]. Here we adopt the simple μ-approximation representation in [21] as $S = \left\{\delta \in R_+^m : \max_{e_{ij} \in E}\{\delta(e_{ij}) + \sum_{e_{uv} \in \Omega(e_{uv})} \delta(e_{uv})\} \leq 1\right\}$, where $\Omega(e_{uv})$ denotes a specific set of links with respect to e_{uv} (see more details in [21]), and $\delta(e_{ij}) = \sum_{f_{pq} \in \mathcal{F}} d_{pq} x_{ij}^{pq}/b_{ij}$. Based on the definition of μ in [21], for a conventional data center network with the physical layout and realistic wireless setting in [16], the μ value is less than 20. In this way, the LP-relaxed problem $\widetilde{\mathcal{P}}_0$ is presented as follows:

$$\begin{aligned} &\min\ \mathcal{C}_{\mathcal{AB}} + \beta \times \mathcal{T}_{\mathcal{B}} \\ &\text{subject to } x_{ij}^{pq} \in [0,1], y_{ij} \in [0,1] && (15)\\ &\qquad \delta(e_{ij}) + \sum_{e_{uv} \in \Omega(e_{ij})} \delta(e_{uv}) \leq 1,\ \forall e_{ij} \in E && (16)\\ &\qquad \text{Constraints}\ \ (1),(2),(8),(9) && (17)\end{aligned}$$

where constraint (15) is the relaxtion of 0-1 variables y_{ij} and x_{ij}^{pq}, and constraint (16) is the relaxtion of IS selection.

C. Decomposition and Rounding

After getting the optimal fractional solution of $\widetilde{\mathcal{P}}_0$, we will decompose and round the solution to 0-1 integer one by one. Despite that introducing configurable links into the data center will provide more flexibility for the network construction, there is a challenge to achieve higher performance taking full advantage of the flexibility. Specifically, it is important to find a right order for the decomposing-and-rounding. As migrating VM will introduce extra cost and reduce the flow performance, we would like to migrate as few VMs as possible, and flexibly

[2]The constraints (8)(9) and $y_{ij} \in \{0,1\}$ define $\{y_{ij}\}$ as an IS, where two variables y_{ij} and y_{pq} have a conflict if the constraints (8)(9) are not satisfied. Thus we can use the *incidence vector* to represent an IS of $\{y_{ij}\}$.

configure the links for data transmissions. Therefore, we will build transmission links after confirming the VM placement.

First, we will decompose and round the fractional VM placement. The main challenge is how to characterize the VM placement efficiently for approximation. The motivation of our technique is modeling a valid VM placement as a *constraint bipartite graph*. After getting the optimal fractional placement $\overline{\mathbf{Y}} = \{y_{ij}: y_{ij} \in [0,1]\}$ by solving $\widetilde{\mathcal{P}}_0$, we use the CCD algorithm (**Algorithm 2**) to perform its convex combination decomposition[3] as follows. Considering an VM or slot as a vertex and the matchings as edges, we can model the one-to-one matching from the VM set to slot set as a bipartite graph G_b. The conflict relationship among the matching edges is defined by constraints (8)(9), and thus we can construct the corresponding conflict graph $\widehat{G}_b$, in which each matching edge that places one VM to one slot is taken as a vertex, and there is an edge between two vertices in $\widehat{G}_b$ if they have a conflict. An IS in $\widehat{G}_b$ exactly denotes an one-to-one matching from VMs to slots, i.e., an VM placement. By calling CCD algorithm with $\widehat{G}_b$ and $\overline{\mathbf{Y}}$ as the input conflict graph and weight vector respectively, we can obtain the output of several VM placements $\{I_i\}$ with corresponding weights $\{w_i\}$.

As Algorithm 2 shows, CCD greedily selects an IS and sets its weight as the minimum weight of links in the IS (line 3-7). Then the link weights are updated and the links with the zero weight is removed (line 9-10). The procedure repeats until no link is left to select. Since the weight of IS is the minimum weight of links in the IS, there is at least one link to be removed from the set Φ at each repetition. Hence the CCD at most performs m loops and the total time complexity is $O(m^2)$, where m is the number of network links.

Algorithm 2 CCD: **C**onvex **C**ombination **D**ecomposition

Input: A conflict graph $G_c(V_c, E_c)$ and a vertex vector $\{x(v) : v \in V_c\}$ where $x(v) \in [0,1]$
Output : The decomposed ISs $\{I\}$ with weights $\{w\}$
1: $\Gamma \leftarrow \emptyset, \Phi \leftarrow \{v \in V_c : x(v) > 0\}$
2: **while** $\Phi \neq \emptyset$ **do**
3: $I \leftarrow \emptyset$
4: **for** $v \in \Phi$ **do**
5: if v does not conflict with nodes in I add v to I
6: **end for**
7: $w \leftarrow min_{v \in I} x(v)$, and add (I, w) to Γ
8: **for** $v \in I$ **do**
9: $x(v) \leftarrow x(v) - w$
10: if $x(v) = 0$, remove v from Φ
11: **end for**
12: **end while**

If the vertex v is selected in Φ with a specific order for line 4 in Algorithm 2, the CCD solves the conventional fractional coloring problem with a total weight $\sum_i w_i \leq 1$ [21]. Finally, we select one VM placement I_i with a probability as its corresponding weight w_i to be the output,[4] which we refer to as the rounding step.

An algorithm is defined to be *ρ-relaxed* if it can achieve a solution within ρ ($\rho \geq 1$) times the optimal solution of the LP relaxation of the integer programming problem. In the following, we show that the above decompose-and-rounding procedure for VM placement selection is a ρ-relaxed algorithm.

First, we illustrate the feasibility of transforming the decomposition of fractional placement to that of its objective value of problem $\widetilde{\mathcal{P}}_0$. This is non-trivial to achieve because the placement variable y_{ij} is coupled with another two sets of variables in the constraints, i.e., the routing variable x_{ij}^{pq} and link variable z_{ij}. Since $\widetilde{\mathcal{P}}_0$ can be solved as a LP problem when given an arbitrary VM placement as its input, we denote the optimal objective value of $\widetilde{\mathcal{P}}_0$ as $\widehat{O}(M)$ with an input placement M. Then we have:

Lemma 1 (Decomposition Feasibility): Let M_f denote the optimal fractional placement by solving $\widetilde{\mathcal{P}}_0$, where M_f can be decomposed as $M_f = \sum_i w_i M_i$. We have that: (a) the integer placement M_i with the same flow routing paths but $1/w_i$ times flow demands that in M_f is a feasible solution of problem $\widetilde{\mathcal{P}}_0$ (denote the corresponding objective value as $\widehat{O}'(M_i)$); (b) $\widehat{O}(M_f) = \sum_i w_i \widehat{O}'(M_i)$.

Proof: See the detailed proof in Appendix B. ■

Based on the above Lemma, suppose there are totally n nodes in the network, we can generate the following theorem for rounding the decomposed placements:

Theorem 3 (Placement Rounding Guarantee): Assume the optimal objective value of $\widetilde{\mathcal{P}}_0$ is $\widetilde{\lambda}_0$. The output objective value is denoted as λ_1^* after rounding the VM placement in $\widetilde{\mathcal{P}}_0$ by the CCD algorithm. Then we have $\lambda_1^* \leq \rho_1 \widetilde{\lambda}_0$ with a high probability, where ρ_1 is $O(\frac{\log n}{\log \log n})$.

Proof: See the detailed proof in Appendix C. ■

Next, we fix the VM placement in $\mathcal{P}_0$ and solve the remaining LP problem. Then we decompose and round the fractional wireless solution $\{x_{ij}^{pq}\}$ to one integer IS. Again, we apply the CCD algorithm to perform the convex combination decomposition, with the wireless conflict graph G_c (Section II-D) and the demand-bandwidth ratio $x(e_{ij}) = \sum_{f_{pq} \in F} d_{pq} x_{ij}^{pq} / b_{ij}$ as the input respectively. Then we select one IS with a probability of its corresponding weight. Now we show that the above decompose-and-rounding procedure for IS selection is a ρ_2-relaxed algorithm:

Theorem 4 (IS Rounding Guarantee): Assume the optimal objective value of the LP problem $\widetilde{\mathcal{P}}_2$ generalized by fixing the VM placement in $\widetilde{\mathcal{P}}_0$ is denoted as $\widetilde{\lambda}_2$. The output objective value after rounding the fractional IS in $\widetilde{\mathcal{P}}_2$ is denoted as λ_2^*. Then we have $\lambda_2^* \leq \rho_2 \widetilde{\lambda}_2$ with a high probability, where ρ_2 is $O(\frac{\log n}{\log \log n})$.

Proof: See the detailed proof in Appendix D. ■

Finally, we fix the VM placement and selected IS and solve the remaining LP problem. We then decompose and round the fractional flows to the single-path integer flows. We apply the *path stripping* method in [27] to decompose the fractional flow $\{x_{ij}^{pq}\}$ to multiple routing paths. The output is the path sets where each routing path is attached with a weight. Similarly, the rounding algorithm for path choice using the

[3]There are many other ways to do the convex combination decomposition of a fractional solution in polynomial time, e.g., the well-known method of solving a linear programming problem in [26]. Here we present an easier one.

[4]If $\sum_i w_i < 1$, we add an empty placement with the weight $1 - \sum_i w_i$ to ensure $\sum_i w_i = 1$. If the empty placement is selected, no migration happens.

path weight as the probability is a ρ_3-relaxed algorithm and ρ_3 is $O(\frac{\log n}{\log \log n})$.

D. Combination of PDR

In this section, we will show how to combine all the approximation ratios that obtained above by PDR together, as one final approximation ratio to the original problem $\mathcal{P}_0$. We first give the lemma of the combination rule as below:

Lemma 2 (Combination Guarantee): Suppose there exists a ρ_1-relaxed, ρ_2-relaxed, ρ_3-relaxed algorithm for the three rounding steps respectively. Then there exists a $(\rho_1\rho_2\rho_3)$-approximation algorithm for $\widetilde{\mathcal{P}}_0$.

Proof: We construct the approximation algorithm for $\mathcal{P}_0$ as follows. First, by rounding the VM placement from $\widetilde{\mathcal{P}}_0$, we have the problem $\mathcal{P}_1$. We use the ρ_1-relaxed algorithm to solve $\mathcal{P}_1$. Denote the output as VM placement $\{y^*\}$ and the output objective as λ_1^*. Since $\widetilde{\lambda}_0$ denote the optimal objective value of $\widetilde{\mathcal{P}}_0$, we have $\lambda_1^* \leq \rho_1\widetilde{\lambda}_0$. By setting the VM placement in $\mathcal{P}_1$ as $\{y^*\}$, we have the LP problem $\widetilde{\mathcal{P}}_2$. Let $\widetilde{\lambda}_2$ as the optimal solution of $\widetilde{\mathcal{P}}_2$, then we have $\widetilde{\lambda}_2 \leq \lambda_1^*$.

Second, by rounding the IS from $\widetilde{\mathcal{P}}_2$, we have the problem $\mathcal{P}_2$. We use the ρ_2-relaxed algorithm to solve $\mathcal{P}_2$. The output IS and objective are respectively denoted as $\{e_{ij}^*\}$ and λ_2^*. Hence we have $\lambda_2^* \leq \rho_2\widetilde{\lambda}_2$. By setting the IS in $\mathcal{P}_2$ as $\{e_{ij}^*\}$, we have the LP problem $\widetilde{\mathcal{P}}_3$. Let $\widetilde{\lambda}_3$ as the optimal solution of $\widetilde{\mathcal{P}}_3$, then we have $\widetilde{\lambda}_3 \leq \lambda_2^*$.

Third, by rounding the splittable flow path, we have the problem $\mathcal{P}_3$. We use the ρ_3-relaxed algorithm to solve $\mathcal{P}_3$. Hence we have $\lambda_3^* \leq \rho_3\widetilde{\lambda}_3$. Therefore, $\lambda_3^* \leq \rho_3\widetilde{\lambda}_3 \leq \rho_3\lambda_2^* \leq \rho_3\rho_2\widetilde{\lambda}_2 \leq \rho_3\rho_2\lambda_1^* \leq \rho_3\rho_2\rho_1\widetilde{\lambda}_0$. This completes the proof. ■

Therefore, we finally generate the following theorem to give the approximation ratio of PDR to the problem $\mathcal{P}_0$:

Theorem 5: Algorithm 1 gives an approximation ratio of $O(\mu\ (\frac{\log n}{\log \log n})^3)$ for problem $\mathcal{P}_0$ with a high probability.

Proof: Let λ_3^* denote the output objective value of $\mathcal{P}_0$ after executing Algorithm 1 and λ_0 denote the optimal objective value of $\mathcal{P}_0$. Our goal is to prove that $\lambda_3^* \leq O(\mu\ (\frac{\log n}{\log \log n})^3)\lambda_0$.

According to Theorem 2, we have $\widetilde{\lambda}_0 \leq \mu\lambda_0$. With lemma 2, we have $\lambda_3^* \leq \rho_3\rho_2\rho_1\widetilde{\lambda}_0$. Hence we have $\lambda_3^* \leq \mu\rho_3\rho_2\rho_1\lambda_0 = O(\mu\ (\frac{\log n}{\log \log n})^3)\lambda_0$. This completes the proof. ■

With three LPs solved in Algorithm 1, its time complexity is polynomial. The VM traffic measurements in real DCNs [6] report that the traffic demands for a large proportion of VMs are relatively stable at large time intervals of several hours. This demonstrates the feasibility of applying our algorithm to achieving a long-term performance benefit on VM communication with infrequent LP solving and topology changes. The VM traffic stability in large time intervals and the polynomial-time complexity of Algorithm 1 allow its good scalability along with the performance guarantee, which avoids the challenge of solving the original integer programming problem at the unscalable exponential time complexity. Recent advances in robust distributed LP solving [28] further show the potential of taking full advantage of the rich distributed computation power in DCNs to run our algorithm efficiently with increasing network scales.

IV. Online Scheduling Design

A. Design Overview

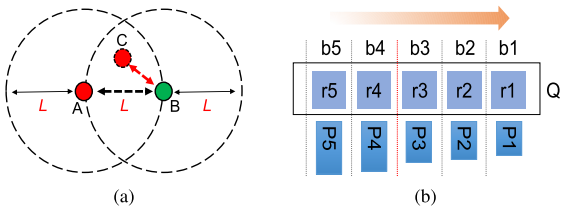
In the previous section, we present PDR to make three challenging decisions for VM migration. However, in order to ensure the performance guarantee, it is relatively time-consuming to solve three LP problems for a large-scale data center network. PDR is better fit for the data centers with long lasting traffic, where the operator can run PDR based on the prediction of flows according to their traffic patterns. As data centers also have many irregular and dynamic flows that cannot be easily predicted, we would like to answer the following key question: *Can we design an online algorithm that can provide migration decisions right after receiving a new flow demand?* The large number of configurable links in the topology-adaptive DCN will lead to an exponential number of topology candidates, which makes it challenging to determine a traffic-aware topology online. Moreover, recent traffic measurements show that the high dynamics in DCN flows make the short-period traffic prediction very challenging [29].

Therefore, we will design an online algorithm that can handle each newly arriving flow in real time. A new flow may trigger the VM migration to achieve a lower total cost. However, the cost would be prohibitive if thousands of flows need VM migration. In addition, each VM migration will affect on-going flows and rescheduling of flows will introduce additional cost. To achieve a lower total cost while guaranteeing the network stability, we will evaluate the effect caused by each newly arriving flow and migrate VMs based on *Lyapunov control theory*.

B. Migration Request Generator

To address the above challenges and avoid frequent VM migrations, we develop an online decision-maker (ODM) to schedule VMs in real-time without traffic prediction. The Lyapunov control theory has been introduced in recent work to give non-prediction-based online scheduling solutions [30]. However, the Lyapunov theory is normally applied in queuing scenarios [31], while our VM migration problem has no natural queuing property. In order to trade off between the VM migration cost and communication cost under the Lyapunov optimization framework, we propose to keep generating proper migration requests with reduced communication cost and maintain them in a queue. This is performed by a module named migration request generator (MRG). Next, to effectively control the migration cost, we design a migration request scheduler (MRS) to schedule the VM migration based on the requests generated by the MRG module. In the following, we will introduce the detailed design of MRG and MRS modules, respectively.

The MRG module is applied to generate migration requests to reduce the communication cost between VMs. To shorten the communication path for each pending VM flow, we search available slots in the intermediate zone between the source VM and the destination VM. Taking Fig. 2a as an example, suppose that there comes a flow from A to B, and the current path length between A and B is L hops. Then we restrict the search space to be within L hops, as determined by two circles (i.e., C_A and C_B) in the figure. As migrating A (or B) just based on the communication requirement of the new coming flow may influence the on-going flows starting from or destining for A (or B) and introduce additional total cost, when we search each available slot in the overlapping



(a) (b)

Fig. 2. Example of online decision-maker. (a) Migration request generator. (b) Migration request scheduler.

region of C_A and C_B, we take into account on-going flows and generate a new placement with a lower total cost. In other words, each generated VM migration request will reduce the total cost of the flows related to the migrated slots. As Fig. 2a shows, we find C in the overlapping region, and generate a request that migrates A to C to achieve a lower total cost in a global view. After the migration, all flows that start from or destine for A will reach C, and A will be recycled to the set of available slots to serve the future flows. After that, we pick wireless links from the available wireless link set and build them between C and B to further reduce the communication cost.

C. Migration Request Scheduler

To schedule the migration requests generated by MRG, we push these requests into a queue and denote it by Q. At each time slot, we have two options to handle each request in Q: *processing it* or *skipping it*. We design a migration request scheduler named MRS to make the final decision. For example, in Fig. 2b, there are five migration requests queued in Q. For each request, there are corresponding migration cost and communication cost that can be used to make migration decision, denoted by b and P. After we have identified the last request r_3 that could satisfy the migration constraints, we process all the requests in Q that are not queued after r_3.

Processing any request q in Q at time t will generate a migration cost (as shown in Section II-E) and we denote it by $q(t)$. For the i-th request, the migration cost is $q_i(t)$. Suppose there are N requests in Q at time t, then the total migration cost (i.e., the queue backlogs) can be computed as:

$$Q(t) \triangleq \sum_{i=1}^{N} q_i(t). \tag{18}$$

At each time slot, there are requests arriving at and leaving from Q. Suppose that MRG generates M requests at time t, we denote $a(t)$ as the total migration cost of the new requests coming at time t, and $b(t)$ as the total cost of requests processed at time t. Let a binary number $x(t)$ denote each request's status, where $x(t) = 0$ means *skip it* and $x(t) = 1$ means *process it* at time t. Then we have:

$$a(t) = \sum_{i=1}^{M} q_i(t) \tag{19}$$

$$b(t) = \sum_{i=1}^{N} q_i(t) \cdot x_i(t) \tag{20}$$

$$Q(t+1) = max[Q(t) + a(t) - b(t), 0] \tag{21}$$

Following the Lyapunov framework in [30] and [32], our *Lyapunov function* is defined as:

$$\mathcal{L}(t) \triangleq \frac{1}{2} Q(t)^2 \tag{22}$$

Then the one-step Lyapunov drift can be computed as:

$$\Delta\mathcal{L}(t) = \mathbb{E}[\mathcal{L}(t+1) - \mathcal{L}(t)|Q(t)] \tag{23}$$

By combining the equations (19)-(23), we have:

$$\Delta\mathcal{L}(t) \leq B(t) - Q(t)\mathbb{E}[b(t)|Q(t)] + Q(t) \cdot a(t) \tag{24}$$

where $B(t) \triangleq \frac{1}{2}(\mathbb{E}[a(t)^2] + \mathbb{E}[b(t)^2|Q(t)])$.

Since the optimization objective is to minimize the total cost of VMs, we consider both of the migration cost and communication cost simultaneously in ODM. In order to minimize the communication cost at time t through VM migration, we define the communication cost as the *penalty function*, denoted by $\mathcal{T}(t)$. Let the communication cost for the i-th request denoted as c_i, then we have:

$$\mathcal{T}(t) \triangleq \sum_{i=1}^{K} c_i \tag{25}$$

where K is the request number to be processed. We add weighed $\mathcal{T}(t)$ into the Lyapunov drift (V is the weight), and get the following *drift-plus-penalty* expression:

$$\Delta\mathcal{L}(t) + V \cdot \mathbb{E}[\mathcal{T}(t)|Q(t)]. \tag{26}$$

Based on the constraint (24) and expression (26), we have the following lemma:

Lemma 3: Suppose the arrive rate $a(t)$ and process rate $b(t)$ are bounded by $\mathbb{A}$ and $\mathbb{B}$, i.e., $a(t) \leq \mathbb{A}$ and $b(t) \leq \mathbb{B}$, and $a(t)$ is i.i.d over time t, the following property holds:

$$\begin{aligned}&\Delta\mathcal{L}(t) + V \cdot \mathbb{E}[\mathcal{T}(t)|Q(t)] \leq B \\ &-\mathbb{E}\{Q(t) \cdot \mathbb{E}[b(t)|Q(t)] - V \cdot \mathcal{T}(t)|Q(t)\} + Q(t) \cdot \mathbb{A}\end{aligned} \tag{27}$$

where $B \leq \frac{1}{2}(\mathbb{A}^2 + \mathbb{B}^2)$.

By applying the dynamics of queue (21) into the Lyapunov drift (23), we can obtain the fact of (27). According to Lyapunov optimization, the optimal migration should take the *drift-plus-penalty* (as shown in (26)) minimization as an objective. Thus the original problem transforms to minimize the right-hand side of (27), and we get the following optimization objective:

$$\max\ \mathcal{O}(t) \triangleq Q(t) \cdot b(t) - V \cdot \mathcal{T}(t) \tag{28}$$

Based on (28), we have the following theorem:

Theorem 6 (The Boundness of ODM): Suppose the arrival rate $a(t)$ is strictly bounded by network capacity region, and the online scheduling decision in (28) is made at each time slot. For any weight $V > 0$, the time-average communication cost $\overline{\mathcal{T}}_\infty$ and time-average migration cost $\overline{Q}_\infty$ satisfy that:

$$\overline{\mathcal{T}}_\infty = \lim_{T\to\infty} \sup \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\mathcal{T}(t)] \leq \Omega + \frac{B}{V} \tag{29}$$

$$\overline{Q}_\infty = \lim_{T\to\infty} \sup \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[Q(t)] \leq \frac{B + V \cdot \Omega}{\varepsilon} \tag{30}$$

where B and ε are positive constants. Ω is the theoretical optimal time-average communication cost.

Proof: See the detailed proof in Appendix E. ■

Based on (28), we design a migration request scheduler, named online decision maker (ODM), to make the migration decision and process migration requests online. We present the details of ODM in **Algorithm 3**. We first run MRG to generate migration requests and updates Q based on (21) on line 1. Then we calculate the migration cost (i.e., the queue backlog) of Q at time t by (18), denoted by $Q(t)$, and initiate a $cursor$ (line 2 and 3). The $cursor$ is used to indicate the last request to be processed. On lines 4-9, we update $cursor$ based on the objective value of (28). The V in (28) is decided by data center operators according to the real traffic pattern. We will

Algorithm 3 ODM: **O**nline **D**ecision-**M**aker

1: Run **MRG** and update Q by (21)
2: Calculate $Q(t)$ by (18)
3: $cursor \leftarrow 0$
4: **for** i from 1 to $|Q|$ **do**
5: Calculate $b_i(t)$, $P_i(t)$, $O_i(t)$ by (20), (25), (28)
6: **if** $O_i(t) > 0$ **then**
7: Update $cursor$ by i
8: **end if**
9: **end for**
10: Process the first $cursor$ requests in Q and calculate $Q(t)$

evaluate the flow performance under different V settings in Section V-B. After that, we process requests according to $cursor$ and update $Q(t)$ (line 10). The total time complexity of ODM is $O(|Q|)$, where $|Q|$ means the migration request number in Q. If the current migration request number in Q is greater than a pre-set threshold, we will empty the Q and restart scheduling. This is because a heavy queue backlog in Q means a long decision time. In the later section V-B, we will show that the flow performance degrades when the number of requests in queue becomes too large.

V. Evaluation

In this section, we present the evaluation of our solutions against other state-of-art VM migration algorithms through simulations and also validate the performance gains of our solutions through testbed experiments.

A. Simulation Setup

For simulations, we use the public traffic trace from two university data centers provided by [29]. We implement a flow-level simulator using the TCP setting in [33]. We use a 3-layer *fat-tree* [34] with 8-port switches as the simulated network topology. To embed realistic wireless settings, we use the 60GHz rectangular waveguide hardware of AINFO Inc¡¯s horn antenna to build our 60GHz wireless antenna pairs for testing the wireless parameters. The measured 60GHz wireless bandwidth at 10 meters is 2.5Gbps and the interference angle of the antenna is 20°. The Rayleigh fading model is applied to simulate the wireless channel dynamics. The physical placement of racks (one wireless radio per rack) follows the previous work [16], i.e., using 24x48 inch rack, 6 feet between two cluster rows and 10 feet between two cluster columns. During the simulations, we select 100 flows from each trace file (there are totally 29 trace files and each file contains about 1 million flow entries in ten minutes) and assigned their sources and destinations to different VMs randomly.

We first compare the flow performance of the proposed PDR and ODM algorithms. PDR is an LP-based offline solution while ODM schedules VM migration requests in real-time. Furthermore, we evaluate the running time of PDR and ODM by conducting simulations under different traffic settings and network scales. The related configurations are listed in Table III.

We study the performance of our PDR algorithm on three aspects: the VM placement, the adaptive topology and the routing strategy. The details of compared strategies are presented in Table II. There are two typical types of algorithm design for VM placement in the literature: (1) the greedy placement that places each VM to a slot to minimize the increment of the current objective cost [10], [35]; (2) the clustering-and-matching strategy that groups VMs and slots into several VM clusters (based on the traffic demand) and slot clusters (based on the communication distance), and then matches the clusters one by one [6], [20]. To compare with our algorithm, without loss of generality, we use the greedy-fill algorithm (named *Greedy*) in [35] and the clustering-and-matching (named *Cluster*) algorithm in [20] to represent the above two typical types of placement algorithms. We utilize the random placement (named *Random*) that randomly maps the VMs to slots to serve as the baseline for VM placement.

TABLE II
Algorithm Comparison on Different Aspects

Algorithm	Range
Placement	Random, Greedy, Cluster, PDR
Topology	Flyway, PDR
Routing	ECMP, PDR

TABLE III
Runtime Comparison of PDR and ODM

Configuration		Runtime (s)	
Server number	Flow number	PDR	ODM
16	10	$0.8 \sim 1$	$0 \sim 0.001$
	100	$8 \sim 10$	
128	10	$3 \sim 7$	$0 \sim 0.001$
	100	$450 \sim 550$	
1024	10	$240 \sim 290$	$0 \sim 0.001$
	100	$3000 \sim 3500$	

For the wireless setup, we compare our wireless solution with the greedy scheme (named *Flyway*) proposed in [17], which builds wireless links between close-by racks that have high capacity. For the routing strategy, we use the most popular DCN routing scheme ECMP (Equal-Cost Multi-Path) [36] to serve as the baseline strategy. For all the comparisons, we equally scale up the flow traffic size of the flow traces to simulate the changes of network loads.

To compare the flow performance of ODM and existing VM migration solutions, we combine VM placement, adaptive topology construction and flow routing together. We compare ODM with Cluster, Greedy and Random on the flow transmission performance. For existing VM placement strategies, we build wireless links by Flyway, and route flows by ECMP. To make a fair comparison, we apply ECMP as the routing strategy in ODM. We evaluate these solutions under the hotspot traffic pattern, which is common in data center networks [14], [37].

We evaluate the performance of ODM by adjusting the weight parameter V in (28) and the flow arrival rates. By adjusting the trade-off parameter V, data center operators could easily control the VM migration frequency. A large weight represents a strict migration constraint, which means VM migration will occur occasionally. Otherwise, more migration requests will be satisfied in the time slot they are generated.

B. Simulation Results

1) Comparison of PDR, ODM and Random: We evaluate the flow completion time, throughput and total cost of PDR, ODM and the baseline algorithm, Random. Fig. 3a shows the distribution of flow completion time. We can see that the flow completion time of PDR is 77.6% and 30% that of ODM

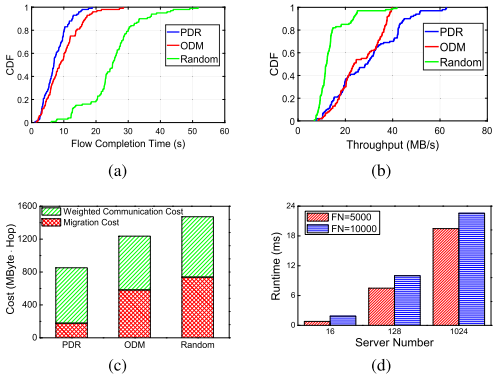


Fig. 3. Flow performance of PDR and ODM. (a) Flow completion time. (b) Flow throughput. (c) Total cost on VM migration and communication. (d) Runtime of ODM.

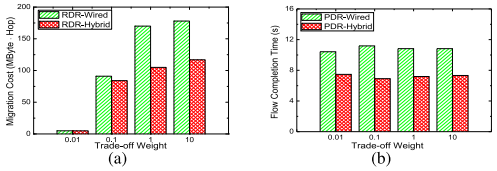


Fig. 4. Performance over different trade-off weights. (a) Migration phase. (b) Communication phase.

and Random on average. In Fig. 3b, the flow throughput of PDR is 38.2% and 111% higher than that of ODM and Random on average. Both PDR and ODM perform much better than Random in terms of the flow performance. In Fig. 3c, we present the total cost (the sum of the migration cost and the weighted communication cost) of PDR, ODM and Random. We can find that, the total cost of PDR and ODM are 42.1% and 16.1% lower than that of Random, respectively. This is because PDR schedules VM migration with the information about future traffic, while ODM makes decisions in real-time without any prediction. Therefore, if the traffic in a data center is predictable in a long period (e.g., the periodic traffic [17]), PDR is a better choice for data center operators.

To evaluate the algorithm running time of PDR and ODM, we run them on a desktop with a 2-core Intel i3-3220 3.3GHz CPU, 16GB memory and a 1TB hard disk under different network settings. We list the network configuration and algorithm runtime in Table III. As the server number or flow number increases, the running time of PDR increases fast, while the time of ODM changes slightly (less than 1ms), and is stable within 1ms. Limited by the running time of PDR, we compare PDR and ODM in a small-scale data center with 100 flows. In Fig. 3d, we also evaluate the running time of ODM when the flow number (FN) equals 5000 or 10000. We can see that, the time of ODM increases with the network scale and flow number, but the running time is kept within several milliseconds. This indicates that ODM can be better used for real-time VM scheduling when the traffic is highly random and dynamic.

2) Trade-Off Performance: As Fig. 4 shows, we evaluate the migration and communication performance by changing the trade-off weight β between the two costs. To evaluate the wireless effects, we compare two versions of PDR: the normal PDR (*PDR-Hybrid*), and the restricted PDR using only the wired links (*PDR-Wired*). In Fig. 4a, we can see that the migration cost for both cases is almost zero when β is smaller than 0.1, but increases when β becomes larger. This is because

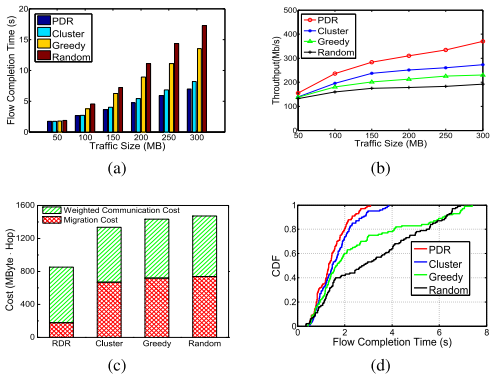


Fig. 5. Performance of different VM placement strategies. (a) Average flow completion time. (b) Average flow throughput. (c) Total cost on VM migration and communication. (d) CDF of flow completion time.

when the weighted communication cost is too small compared to the migration cost, PDR would ignore the objective of minimizing communication cost. When β is larger than 0.1, the cost of the *PDR-Wired* increases quickly to a large value, which is more than 1.5 times that of the *PDR-Hybrid*. The migration cost of *PDR-Hybrid*, on the other hand, increases very slowly with the weight parameter. The wireless links are utilized by PDR to construct a proper topology to reduce the migration cost, which at the same time, achieves the smaller flow completion time in the communication phase as Fig. 4(b) shows. Since the communication performance is similar for PDR with different weights, without loss of generality, we use $\beta = 0.1$ as the default weight value of PDR in the following.

3) Performance of VM Placement: In Fig. 5, we evaluate different VM placement solutions. We compare PDR and ODM with existing placement strategies, respectively. In the comparison between PDR and existing solutions, to make a fair comparison, all the compared solutions use exactly the same wireless setup as PDR, and also compute their routing paths by solving the remaining flow routing LP problem as that done by PDR. In Fig. 5a and Fig. 5b, the PDR achieves the lowest completion time and the highest throughput for all the network loads, while its performance gain compared to others becomes larger with a higher load. This demonstrates that the placement of PDR works the best with the underlying adaptive topology to handle the congested flows. For both figures, the Random solution performs the worst as it is not aware of any resource usage. The Cluster and Greedy perform very similarly when the network load is light, while the Cluster outperforms Greedy when the load turns heavy. This indicates that the greedy placement algorithm works more effectively in a non-congested network.

The Fig. 5c and Fig. 5d show the detailed performance of PDR and four existing solutions. In Fig. 5c, we can see that the total migration costs of Cluster, Greedy and Random are similar, which is about 4 times that of PDR. At the same time, the communication cost of all the solutions are similar while the communication cost of Cluster is a little smaller than that of PDR. This is because the Cluster optimizes the communication cost only without the consideration of the migration cost, while the PDR sacrifices some communication cost to gain a much lower migration cost. As Fig. 5d shows, the maximum flow completion time of PDR is about 20% less than that of others. This should be attributed to the topology-aware VM placement in PDR, while other placements fail to exploit the adaptive topology.

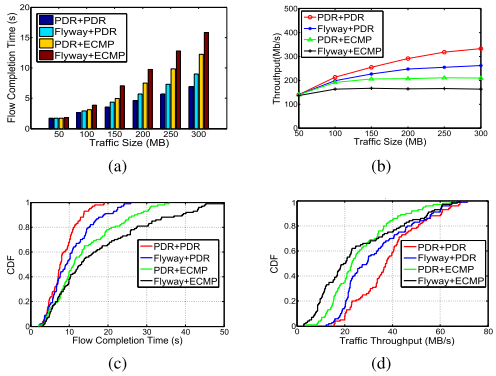


Fig. 6. Flow performance of different wireless setup and routing. (a) Average flow completion time. (b) Average flow throughput. (c) CDF of flow completion time. (d) CDF of flow throughput.

4) Performance of Wireless Setup and Routing: In Fig. 6, we evaluate two wireless setups (PDR and Flyway) and two routing strategies (PDR and ECMP) together. For example, we denote the solution using the wireless setup of PDR and the ECMP routing as *PDR+ECMP*. To make a fair comparison, all the solutions use the same VM placement as PDR. In Fig. 6a and Fig. 6b, we can see that PDR performs the best on both the flow completion time and throughput over various network loads, with its concurrent consideration of the placement and routing in forming the topology. The Flyway solution, however, adapts the topology based on the placement only, without considering the flow routing paths. This difference in wireless setup brings PDR a 25% higher throughput than Flyway when the network load is high.

Another interesting finding is that *PDR+ECMP* outperforms *Flyway+ECMP* on both the flow completion time and throughput. It indicates that even though using the same ECMP routing, the topology built by the PDR cooperates better with the VM placement than the topology built by a greedy algorithm. This difference in adapting the topology brings *PDR+ECMP* a 28% throughput improvement compared to *Flyway+ECMP*. The flow details in Fig. 6c and Fig. 6d show that the maximum completion time of 90% flows in PDR is about 30% less than that of other solutions. Moreover, in PDR there are only 20% *slow flows* that have throughputs less than 25MBps, while there are at least 40% such flows in the others.

5) Comparison of ODM and Existing Solutions: In Fig.7, we compare ODM and existing solutions when there exits different percentage of hot nodes. We consider the hotspot model in [38], where in addition to the basic simulation setup, a subset of VMs have higher flow arrival rates and larger flow sizes. We conduct evaluations under different hotspot traffic patterns, the number in X-axis means the percentage of hot nodes, the size of flow from hot nodes is about 10x that of normal flows.

As Fig. 7a shows, with the increase of hot nodes, the flow completion time of ODM increases much slower than other three solutions. When there are 30% hot nodes, the flow completion time of ODM is just a quarter that of Random, and about 30% that of Cluster and Greedy. The throughput performance in Fig. 7b shows a similar trend. When the fraction of hot node increases from 0% to 30%, the throughput of ODM declines slightly (<10%), while it declines sharply (about 50%) for other solutions. We can see that, benefited from the migration decisions derived from Lyapunov

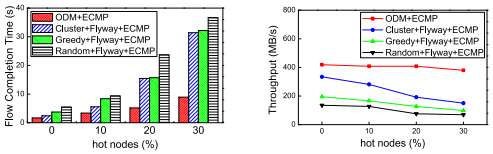


Fig. 7. Performance of ODM and existing solutions. (a) Flow completion time. (b) Flow throughput.

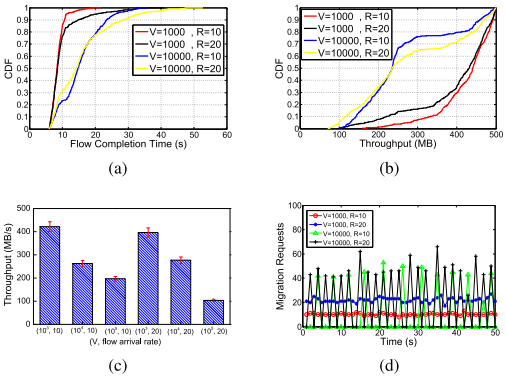


Fig. 8. Flow performance of ODM. (a) CDF of flow completion time. (b) CDF of flow throughput. (c) Average flow throughput. (d) Request number.

optimization, ODM can handles hotspots much better than other existing solutions.

6) Performance of ODM: As Fig. 8 shows, we evaluate the flow performance of ODM under different traffic and parameter settings. The trade-off parameter V is used to balance the migration cost and communication cost. In our problem, it also controls the VM migration frequency. The parameter R represents flow arrival rate. First, we find that the trade-off parameter V should not be set too large. As Fig. 8a shows, when V becomes large, the flow completion time increases about 50% on average. In Fig. 8b and Fig. 8c, we can see the downtrend of throughput with the increase of V. When we fix the flow arrival rate, increasing V decreases the throughput, but the drop speed slows down gradually. On the other hand, we can see that a smaller V will bring higher migration cost. Fig. 8d shows the relationship between migration frequency and network settings. When V is set as 10^3, VM migration occurs in each time slot, while a larger V reduces the migration frequency thus the migration cost. Due to the balance effect of V, when optimizing the flow transmission performance using ODM, data center operators can adopt a proper V to control the VM migration cost based on their requirements.

C. Experiment Results

In this section, we present a hardware testbed to validate the performance gains of our solutions, PDR and ODM. Our testbed consists of three hosts with the 4-core Intel Xeon CPU and 8GB RAM. Two hosts (called *client*) are used for the virtualization of the hosts and switches, while the left host is running as the SDN controller implemented by *Floodlight v1.1* [39]. We use *VirtualBox v4.1.12* [40] to implement the virtual servers and use *OpenvSwitch v1.4.6* [41] to implement the virtual switches. The clients send flow statistics to the controller and also receive the commands from it to deploy the routing entries at the virtual switches. We build a partial 3-layer fat-tree topology using 4-port virtual switches (see Fig. 9).

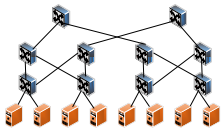


Fig. 9. Topology of testbed.

Trace	Ave. R (s)	Std. R	Ave. S (MB)	Std. S	Burst
1	17.5	14.8	1302	262	NO
2	5	5	1580	390	YES
3	3.33	4.7	1585	515	YES
4	10	0	1745	383	NO
5	6.67	9.4	1786	402	NO
6	10	8.2	2542	1110	YES

Fig. 10. Analysis of flows.

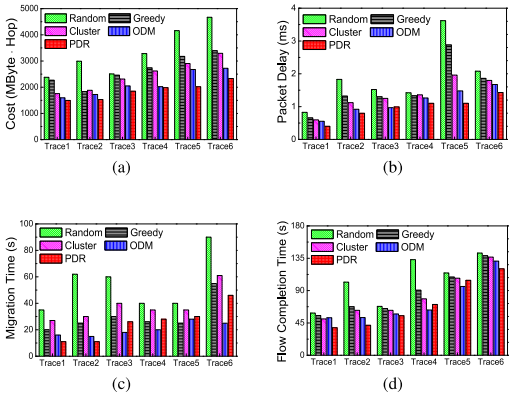


Fig. 11. Flow performance in the testbed. (a) Total cost on VM migration and communication. (b) Packet delay. (c) Total migration time. (d) Flow completion time.

Since current OpenvSwitch does not support wireless transmission, we use wired link to simulate the wireless link and set the wireless rate based on the transmission distance and the Rayleigh-fading effect following our measurements using 60GHz wireless antennas introduced in the simulation setup. Since our algorithm only builds the non-interfered wireless links, this approximation is acceptable as the bandwidth changes due to interference can be ignored. In the experiment, we find that the OpenvSwitch can only achieve full transmission rate up to 100MBps. Without loss of generality, we equally scale down the bandwidth setting of wired links and wireless links to tens of MBps in our testbed based on the link bandwidths in a DCN [16]. For each round of experiment, we randomly derive four flows from a different trace file (totally six files from *Trace1* to *Trace6*) which contains about one million flow entries. We list the characteristics of flows in each round of experiment in Fig. 10, including the average arrival rate (Ave. R), the standard deviation of arrival rate (Std. R), the average flow size (Ave. S), the standard deviation of flow size (Std. S). We also indicate if a flow contains the traffic burst, "YES" if there exists a burst and "NO" otherwise. In our experiments, the VM size to migrate is set as 100MBytes. The controller records the flow rates and completion time, and we also monitor the average packet delay on the clients. We will present the experiment results under different flow characteristics in the following.

We first compare the total cost of different algorithms in Fig. 11a. Among all the five algorithms, PDR achieves the lowest total cost, which is consistent with the simulation results. Although the total cost of ODM is higher than PDR, it is lower than other three algorithms (Random, Greedy and Cluster) in our experiments. As Fig. 11b shows, for all the traces, the flows in PDR have an average packet delay lower than that of Random, Greedy, Cluster and ODM, and the difference are about 48.5%, 34.1%, 27.7% and 15.2% on average. PDR generates a better VM placement that can cooperate well with the adapted topology to cut short the routing paths, which helps avoid going through the congestion nodes and thus reduce the communication delay. The Random solution, however, generalizes a random VM placement which pays a high cost on the communication. We present the detailed VM and flow performance in Fig. 11c and Fig. 11d. In Fig. 11c, we can see that the total migration time, which is consist of the downtime (tens of milliseconds) and the migration time (several seconds) [4]. The migration time of ODM and PDR is smaller than that of other three algorithms. Compared with Random, ODM and PDR reduce the total migration time by 62.7% and 53.5% on average. Taking the migration cost into account, ODM and PDR schedule VMs based on traffic requirements and overall cost. They migrate fewer VMs than Random in general. In Fig. 11d, the flows in ODM and PDR achieve a smaller completion time compared to other algorithms in most cases. Compared with Random, ODM and PDR reduce the flow completion time by about 26.9% and 30.4% on average. From the above, the experimental results demonstrate the effectiveness of our solutions to ensure the high performance of all the flows in a real testbed.

VI. Deployment Discussion

A. Solution Selection

To optimize VM migration in topology-adaptive DCN, we have proposed two different solutions, PDR and ODM. Despite that PDR outperforms ODM in general flow performance, it requires the traffic prediction. It could fit the data centers that have a clear traffic pattern, e.g., the periodic pattern [17]. Compared with PDR, ODM performs better on algorithm running time. It generates and schedules VM migration requests in real-time without any prediction for future flows. In practice, data center operators can select one of the above solutions according to their needs.

B. Application of OCSs and FSOs

We show how to extend our model to the case of using OCSs [13], [19] or FSOs [11], [14] in DCNs. The OCS changes the topology by solving a bipartite matching from its input ports to the output ports [19]. In this way, the OCS links are built with respect to the bipartite matching polytope, which is known to have an exact relaxation as its convex hull [24]. Therefore, the PDR solves it with an approximation ratio improved by a constant μ compared to the wireless mechanism. The FSO is a special case of wireless radio that has little interference footprint, but has a link conflict that each FSO port can build at most one link [14]. Hence it can also be modeled as the independent set and the PDR solves it with the same approximation ratio.

C. Application of SINR Model

We show that although we use an binary interference model for a general optimization purpose of the topology-adaptive DCN, our solution can also be easily extended to solve the SINR interference model. Actually, the work in [25] proves that the inductive independence number μ is bounded by $\log(n)$ for the edge-weighted conflict graph described by the SINR interference model, where our PDR solution solves it with an approximation ratio $O(\log^4 n/\log\log^3 n)$.

D. Migration Phase vs. Communication Phase

As hardware upgrades quickly, VM consolidation is common in data centers, e.g., a physical server can host more than 60 VMs [42]. As different services running in data centers generate various traffic patterns and huge amounts of flows with different service requirements (e.g., SLAs [43]), VM migration becomes essential and frequent, thus generating non-ignorable migration traffic. In order to satisfy different requirements, providing performance guarantees by balancing the migration cost and communication cost through the weight-balancer is necessary and efficient.

VII. Related Work

A. Hybrid Data Center Networking

Recent years, there are mainly two technologies to achieve the topology-adaptive objective in DCNs. The first is adding the 60GHz wireless radios or the Free-Space Optics (FSO, which is another wireless technology that differs from 60GHz) to build configurable wireless links [11], [12], [14], [16], [17]. The second is adding the optical circuit switches (OCS), which has the ability of fast circuit switching to adapt the topology [13], [15], [18], [19]. Researchers propose to construct hybrid DCNs with these technologies to make the adaptive-topology DCNs into reality. *Flyway* is the first work that proposes to apply 60GHz wireless technology in DCNs [17]. To overcome the easy-blocking problem, Zhou *et al.* [16] first take ceiling mirrors as reflecting medium to avoid signal blocking and transmit data flows efficiently. Ghobadi *et al.* [11] explore FSO-based interconnections in data center, which enables the direct links between all pairs of racks. Besides wireless technologies, some researchers leverage circuit switching technology to build hybrid DCNs. Chen *et al.* [44] propose an optical switching architecture for DCNs to achieving full flexibility. XFabric reconfigures topology and uplink placement using a set of independent small circuit switches in in-rack networks, efficiently coping with partial reconfigurability [45]. The focus of the above studies is on the design of the hybrid network architecture in the data center, while in this paper we take advantage of the reconfigurable links to optimize the VM migration in topology-adaptive DCN.

B. VM Migration in Data Centers

Employing VMs in data centers is a common practice and can bring many benefits, such as improved resource utilization, load balancing and fault tolerance. Barham *et al.* [46] show that several VMs running on a physical machines perform well in utilizing physical resources. Birke *et al.* [42] provide a multi-faceted analysis of virtualization in production data center and show its efficiency.

In order to improve the performance of VMs and reduce the traffic cost, VM migration is necessary and efficient for data centers. Clark *et al.* [3] propose the *live migration* mechanism to reduce the performance loss brought by VM migration. Zhang *et al.* [4] and Wang *et al.* [5] propose to optimize the migration time and service downtime through bandwidth allocation. Recent effort in [8], [9] further take the routing options into account and propose to optimize VM migration by placing VMs properly and selecting routing paths for VM communications. Different from the previous studies which focus on VM migration in a static network topology, we study the VM migration with a reconfigurable network topology and consider the joint optimization of migration phase and communication phase. As intensive communications increase quickly in DCNs, VM migration is a good option for data center operators to optimize flow scheduling and make full use of the available resources (e.g., CPU, memory and bandwidth).

VIII. Conclusion

In this paper, we propose a novel paradigm of topology-adaptive DCN to facilitate more efficient VM migration. We show that the ability of flexibly reconfiguring the DCN topology can be exploited to break through the performance bottleneck due to the previous difficulty in trading-off the migration cost and communication cost. We formulate the problem of jointly minimizing the migration cost and communication cost in an adaptive network topology, and show its NP-hardness. For data centers with different flow characteristics, we develop PDR and ODM to schedule the VM migrations periodically and in real-time, respectively. PDR can solve the optimization problem in polynomial time with a proved approximation ratio. It can be applied in data centers where flow demands can be predicted based on history information. ODM schedules the VM migration requests in real-time, which is applicable to run in the data centers with highly random and dynamic flows. We further present the theoretical analysis on the performance bound of ODM. The real-trace based simulations and experiments demonstrate the advantage of our solutions in achieving a smaller flow completion time while ensuring a much lower VM migration cost compared to other state-of-art solutions.

References

[1] V. Medina and J. M. García, "A survey of migration mechanisms of virtual machines," *ACM Comput. Surv.*, vol. 46, no. 3, p. 30, 2014.

[2] B. Hu, S. Chen, J. Chen, and Z. Hu, "A mobility-oriented scheme for virtual machine migration in cloud data center network," *IEEE Access*, vol. 4, pp. 8327–8337, 2016.

[3] C. Clark *et al.*, "Live migration of virtual machines," in *Proc. NSDI*, 2005, pp. 273–286.

[4] J. Zhang, F. Ren, and C. Lin, "Delay guaranteed live migration of virtual machines," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 574–0582.

[5] H. Wang, Y. Li, Y. Zhang, and D. Jin, "Virtual machine migration planning in software-defined networks," in *Proc. IEEE INFOCOM*, Apr. 2015, pp. 487–495.

[6] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proc. INFOCOM*, 2010, pp. 1–9.

[7] A. Munir, T. He, R. Raghavendra, F. Le, and A. X. Liu, "Network scheduling aware task placement in datacenters," in *Proc. 12th Int. Conf. Emerg. Netw. EXperim. Technol. (CoNEXT)*, New York, NY, USA, 2016, pp. 221–235. [Online]. Available: http://doi.acm.org/10.1145/2999572.2999588

[8] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint VM placement and routing for data center traffic engineering," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2876–2880.

[9] L. Wang, F. Zhang, A. V. Vasilakos, C. Hou, and Z. Liu, "Joint virtual machine assignment and traffic engineering for green data center networks," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 3, pp. 107–112, 2014.

[10] V. Shrivastava *et al.*, "Application-aware virtual machine migration in data centers," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 66–70.

[11] M. Ghobadi *et al.*, "ProjecToR: Agile reconfigurable data center interconnect," in *Proc. SIGCOMM*, 2016, pp. 216–229.

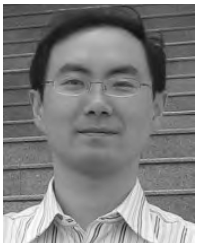
[12] Y. Cui *et al.*, "Diamond: Nesting the data center network with wireless rings in 3D space," in *Proc. NSDI*, 2016, pp. 657–669.

[13] H. Liu *et al.*, "Scheduling techniques for hybrid circuit/packet networks," in *Proc. CoNEXT*, 2015, p. 41.

[14] N. Hamedazimi *et al.*, "FireFly: A reconfigurable wireless data center fabric using free-space optics," in *Proc. SIGCOMM*, 2014, pp. 319–330.

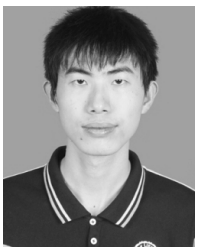
[15] G. Porter *et al.*, "Integrating microsecond circuit switching into the data center," in *Proc. SIGCOMM*, 2013, pp. 447–458.

[16] X. Zhou *et al.*, "Mirror mirror on the ceiling: Flexible wireless links for data centers," in *Proc. SIGCOMM*, 2012, pp. 443–454.
[17] D. Halperin, S. Kandula, J. Padhye, P. Bahl, and D. Wetherall, "Augmenting data center networks with multi-gigabit wireless links," in *Proc. SIGCOMM*, 2011, pp. 38–49.
[18] G. Wang *et al.*, "c-Through: Part-time optics in data centers," in *Proc. SIGCOMM*, 2010, pp. 327–338.
[19] N. Farrington *et al.*, "Helios: A hybrid electrical/optical switch architecture for modular data centers," in *Proc. SIGCOMM*, 2010, pp. 339–350.
[20] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Comput. Netw.*, vol. 57, no. 1, pp. 179–196, 2013.
[21] P.-J. Wan, "Multiflows in multihop wireless networks," in *Proc. MobiHoc*, 2009, pp. 85–94.
[22] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman, 1979, pp. 61–62.
[23] S. Sen, D. Shue, S. Ihm, and M. J. Freedman, "Scalable, optimal flow routing in datacenters via local link balancing," in *Proc. CoNEXT*, 2013, pp. 151–162.
[24] *Matching Polytope*. Accessed: Jan. 28, 2010. [Online]. Available: http://www.imsc.res.in/meena/matching/polytope
[25] M. Hoefer, T. Kesselheim, and B. Vöcking, "Approximation algorithms for secondary spectrum auctions," *ACM Trans. Internet Technol.*, vol. 14, nos. 2–3, p. 16, 2014.
[26] R. Carr and S. Vempala, "Randomized metarounding," in *Proc. STOC*, 2000, pp. 58–62.
[27] P. Raghavan and C. D. Tompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, 1987.
[28] D. Richert and J. Cortés, "Robust distributed linear programming," *IEEE Trans. Autom. Control*, vol. 60, no. 10, pp. 2567–2582, Oct. 2015.
[29] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. IMC*, 2010, pp. 267–280.
[30] M.-R. Ra *et al.*, "Energy-delay tradeoffs in smartphone applications," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Ser.*, 2010, pp. 255–270.
[31] M. J. Neely, "Energy optimal control for time-varying wireless networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 7, pp. 2915–2934, Jul. 2006.
[32] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.
[33] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. NSDI*, 2010, p. 19.
[34] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. SIGCOMM*, 2008, pp. 63–74.
[35] D. Erickson, B. Heller, N. McKeown, and M. Rosenblum, "Using network knowledge to improve workload performance in virtualized data centers," in *Proc. IEEE IC2E*, Mar. 2014, pp. 185–194.
[36] C. Hopps, "Analysis of an equal-cost multi-path algorithm," document RFC 2992, IETF, 2000.
[37] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas. Conf.*, 2009, pp. 202–208.
[38] X.-Y. Li and Y. Wang, "Simple approximation algorithms and PTASs for various problems in wireless ad hoc networks," *J. Parallel Distrib. Comput.*, vol. 66, no. 4, pp. 515–530, 2006.
[39] *Floodlight SDN Controller*. [Online]. Available: http://www.projectfloodlight.org/floodlight/
[40] *Virtual Box*. [Online]. Available: https://www.virtualbox.org/
[41] *Open vSwitch*. [Online]. Available: http://openvswitch.org/
[42] R. Birke, M. Björkqvist, C. Minkenberg, M. Schmatz, and L. Y. Chen, "When virtual meets physical at the edge: A field study on datacenters' virtual traffic," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 43, no. 1, pp. 403–415, 2015.
[43] P. Patel, A. H. Ranabahu, and A. P. Sheth. (2009). *Service Level Agreement in Cloud Computing*. [Online]. Available: http://corescholar.libraries.wright.edu/knoesis/78
[44] K. Chen *et al.*, "OSA: An optical switching architecture for data center networks with unprecedented flexibility," *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 498–511, Apr. 2014.
[45] S. Legtchenko *et al.*, "XFabric: A reconfigurable in-rack network for rack-scale computers," in *Proc. 13th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2016, pp. 15–29.
[46] P. Barham *et al.*, "Xen and the art of virtualization," *ACM SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, Dec. 2003.



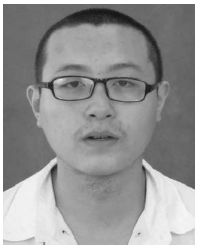
Yong Cui received the B.E. and Ph.D. degrees in computer science and engineering from Tsinghua University, China, in 1999 and 2004, respectively. He is currently a Full Professor with the Computer Science Department, Tsinghua University. He has authored over 100 papers in the refereed conferences and journals with several best paper awards. He has coauthored seven Internet standard documents (RFC) for his proposal on IPv6 technologies. His major research interests include mobile cloud computing and network architecture. He is currently a Working Group Co-Chair in the IETF. He served or serves at the editorial boards on the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, the IEEE TRANSACTIONS ON CLOUD COMPUTING, and the *IEEE Internet Computing*.

Zhenjie Yang received the B.E. degree in networking engineering from the Dalian University of Technology, Liaoning, China, in 2015. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include data center networking and cloud computing.



Shihan Xiao received the B.Eng. degree in electronic and information engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2012. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University, Beijing. His research interests are in the areas of data center networking and cloud computing.

Xin Wang received the B.S. and M.S. degrees in telecommunications engineering and wireless communications engineering from the Beijing University of Posts and Telecommunications, Beijing, China, and the Ph.D. degree in electrical and computer engineering from Columbia University, New York, NY, USA. She was a member of Technical Staff in the area of mobile and wireless networking at Bell Labs Research, Lucent Technologies, NJ, USA, and an Assistant Professor with the Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY. She is currently an Associate Professor with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY. Her research interests include algorithm and protocol design in wireless networks and communications, mobile and distributed computing, as well as networked sensing and detection. She received the NSF Career Award in 2005 and the ONR Challenge Award in 2010. She has served in executive committee and technical committee of numerous conferences and funding review panels, and serves as an Associate Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING.



Shenghui Yan received the B.S. degree from the Computer Science and Technology Department, Beihang University, Beijing, China, in 2014, and the M.S. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, in 2017. His research interests are in the areas of data center networking.