Traffic-aware Virtual Machine Migration in Topology-adaptive DCN

Shihan Xiao*, Yong Cui*, Xin Wang†, Zhenjie Yang*, Shenghui Yan* and Liu Yang‡
*Dept. of Computer Science and Technology, Tsinghua University, Beijing, China
†Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, New York, USA
‡Department of Computer Science, Beijing University of Posts and Telecommunications, China
xiaoshihan.xsh@gmail.com, cuiyong@tsinghua.edu.cn, x.wang@stonybrook.edu, yangzhenjie.dlut@gmail.com

***Abstract*—Virtual machine (VM) migration is a key technique for network resource optimization in modern data center networks (DCNs). 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. When the VM demands change, however, a great cost will be paid on the VM migration. With the advance of software-defined network (SDN), recent studies have shown great potential to implement an adaptive network topology at a low cost. Taking advantage of the topology adaptability, in this paper, we propose a new paradigm for VM migration by dynamically constructing a topology 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. Then we develop a novel progressive-decompose-rounding (PDR) algorithm to solve this problem in polynomial time with a proved approximation ratio. Extensive trace-based simulations show that PDR can achieve higher flow throughput among VMs with only 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 solution.**

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 balance, energy efficiency, etc [1].

Intensive recent efforts have been made to minimize the cost of migrating one or several VMs from an initial placement to another given destination placement, namely the **migration phase**. Clark et al. [2] first present a systematic *live migration* mechanism with practically small service downtime for a single VM migration. Recent work [3], [4] further proposes to optimize the migration time and service downtime by proper bandwidth allocation for the live migration. Benefited from the advantage of high-performance live migration, there comes another research theme to reduce the communication cost of VMs under the optimized destination placement, which we refer as the **communication phase**. With the increasing trend of running communication-intensive applications in DCNs [5], 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 [5] 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 [6], [7] further take the routing options into account and propose to jointly optimize the VM placement and proper routing paths for the VM communication.

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 challenge between these two costs, e.g., a *good* placement obtained by minimizing the communication cost may be a *bad* option for the migration, due to the high cost of reshuffling existing VMs [8]. To address this challenge, our basic motivation is that, if the network topology can be changed dynamically with reconfigurable links, the servers who 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 efficiently.

Therefore, in this paper, we fully explore a novel paradigm, called the *topology-adaptive DCN*, to lower the cost of both the migration phase and communication phase easily. With the advance of SDN technologies in DCN, recent studies have shown the great potential to implement a reconfigurable network topology [9]–[17]. 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, which is another wireless technology that differs from 60GHz) to build configurable wireless links [9], [10], [12], [14], [15]. The second is adding the optical circuit switches (OCS), which has the ability of fast circuit switching to adapt the topology [11], [13], [16], [17]. The core of the above technologies is to collect traffic information through the OpenFlow protocol in the SDN platform and then build *configurable links* on-demand by the SDN controller [12]. Since the topology is only updated based on the traffic demands at every period of large timescales (e.g., several hours [5]), the relatively small delay on switching the topology (about several microseconds [13], [15]) would have little impact on the total performance gains [14], [15].

Recent studies on the topology-adaptive DCN [9]–[17] generally focus on the basic network requirements, such as the feasibility of new technology and architecture design. In this work, we take the initiative to explore how much benefit up-layer applications could gain with the topology-adaptive

978-1-5090-3281-5/16/$31.00 ©2016 IEEE

paradigm. Without loss of generality, we take the wireless mechanism as the main analysis reference in our paper. Then we will show that adapting topology through OCS or FSO is a special case of the wireless solution (see Section III-E).

Specifically, to minimize the total cost in the VM migration and communication, we will jointly optimize three challenging decisions 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; (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. In each period, our system takes the traffic demands among VMs as the input and updates all the above three decisions to minimize the total cost of VM migration and communication.

To the best of our knowledge, this is the first work to study the VM migration with a reconfigurable network topology. Moreover, existing work on the joint optimization of the VM placement with other metrics like routing, link utilization or energy consumption [5]–[7], [18] 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 jointly consider the costs of migration phase and communication phase, and formulate the traffic-ware VM migration problem in an adaptive topology, which is shown to be NP-hard.
- We propose a novel progressive-decomposition-rounding (PDR) algorithm to solve the migration problem and prove its approximation ratio. 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.
- We conduct real-trace based evaluations to demonstrate the efficiency of our solution under various scenarios, and then validate its feasibility of improving the flow performance with implementation over an OpenvSwitch-based testbed.

The remainder of this paper is organized as follows. Section II introduces our problem formulation. In Section III, we provide scheduling analysis and algorithm design. We evaluate the performance of our scheduling algorithm in Section IV, and finally conclude our work in Section V.

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 two

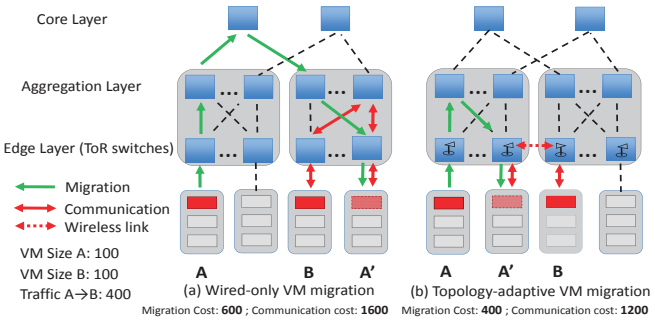


Fig. 1. Cost comparison of different VM migration solutions

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 so 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. In Fig. 1, if VM A and B communicate without any migration, the migration cost is zero and the communication cost is $400\text{MB} \times 6$ hops$=2400$. With an optimal migration performed over a static wired topology, as shown in Fig. 1(a), 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, a simple topology-adaptive solution in Fig. 1(b) achieves both a lower migration cost and communication cost, with the total cost reduced by 33% to gain the performance breakthrough. Finally, the optimal topology-adaptive solution in Fig. 1(b) is to build a direct wireless link between the ToR (Top-of-Rack) switches of A and B, which makes the communication cost the lowest and migration cost zero, i.e., reducing 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 that 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 $f_{ij} \in \mathcal{F}$ from the VM M_i to M_j with a traffic demand d_{ij} ($\mathcal{F}$ 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 [5]. 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:

$\forall v_i \in S, f_{pq} \in \mathcal{F}$:

$$\sum_{e_{ij} \in E} x_{ij}^{pq} - \sum_{e_{ji} \in E} x_{ji}^{pq} = \sum_{k \in \widehat{L}(i)} y_{pk} - \sum_{k \in \widehat{L}(i)} y_{qk} \tag{1}$$

$\forall v_i \in V \setminus S, f_{pq} \in \mathcal{F}$:

$$\sum_{e_{ij} \in E} x_{ij}^{pq} - \sum_{e_{ji} \in E} x_{ji}^{pq} = 0 \tag{2}$$

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. Wireless interference

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 III-E.

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 [19], and we will show how to extend our solution to the SINR interference model in Section III-E. In the binary interference model, two wireless 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 wireless 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.

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 wireless links in G is to select an IS in the conflict graph G_c, i.e., we have

$$z_{ij} \in \{0, 1\}, \ \forall e_{ij} \in E \tag{3}$$

$$z_{ij} + z_{uv} \leq 1, \ \forall (e_{ij}, e_{uv}) \in E_c \tag{4}$$

where z_{ij} denotes whether the link e_{ij} is selected to build[1], and constraint (4) ensures the independence among links.

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:

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

E. Migration and communication cost

Migration cost. To evaluate the cost of migration phase, we use a_{ij} as a fixed value to denote the general migration cost by migrating the VM M_i from its previous slot L_i 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 [4].

[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.

Therefore, the total network cost of the migration phase from placement $\mathcal{A}$ to a new placement $\mathcal{B}$ can be computed as

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

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 [5]. Then the cost of VM communication can be described by the product of the traffic demand and communication distance.

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

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

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 $\mathcal{P}_0$:

$$\min \quad \mathcal{C}_{\mathcal{AB}} + \beta \times \mathcal{T}_{\mathcal{B}}$$

$$\text{subject to} \quad \sum_{v_j \in \mathcal{L}} y_{ij} = 1, \ \forall M_i \in \mathcal{M} \tag{8}$$

$$\sum_{M_i \in \mathcal{M}} y_{ij} \leq 1, \ \forall L_j \in \mathcal{L} \tag{9}$$

$$x_{ij}^{pq} \in \{0,1\}, y_{ij} \in \{0,1\} \tag{10}$$

$$\text{Constraints} \ \ (1),(2),(3),(4),(5) \tag{11}$$

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 wireless interference and link capacity.

By solving this optimization problem, the controller outputs three sets of decisions: (1) the new VM placement $\mathcal{B} = \{y_{ij}\}$; (2) the wireless links $\{z_{ij}\}$ selected to build; (3) the routing paths selected for VM-level flows $\{x_{ij}^{pq}\}$ under the placement $\mathcal{B}$. The related notations and definitions are listed in Table I. Since the general binary integer problem is NP-hard [20], 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 wireless 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 [21].

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 detailed proof in our technical report [22]. ■

III. Scheduling Analysis and 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.

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}

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

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 inequation (5), we can see that the expression of x_{ij}^{pq} is limited by an upper bound as z_{ij}. On the other hand, 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) by characterizing the solution space of x_{ij}^{pq} with that of 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. Since each incidence vector represents an integer point in the vector space of $\{z_{ij}\}$, 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 $\{z_{ij}\} \in \overline{P}$, 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 Q within a constant ratio μ, i.e., $Q \subseteq P \subseteq \mu Q$, 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:

$$\min \quad O(X, Y)$$

$$\text{subject to} \quad X \in Q \tag{12}$$

$$Y \in U \tag{13}$$

$$W(X, Y) = 0 \tag{14}$$

where X denotes the vector of variables $\{x_{ij}^{pq}\}$, and Y denotes the vector of variables $\{y_{ij}\}$. The polytope Q 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 $X \in \mu Q$, $Y \in \mu U$ 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 $\overline{U}$ [23]. 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 $\overline{H} \subseteq Conv(\overline{P}) = P \subseteq \mu Q$ 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 Q [19], [24]. Here we adopt the

[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 $\{y_{ij}\}$.

simple μ-approximation representation in [19] as $Q = \{\delta \in R_+^m : \max_{e_{ij}\in E}\{\delta(e_{ij}) + \sum_{e_{uv}\in\Omega(e_{uv})}\delta(e_{uv})\} \leq 1\}$, where $\Omega(e_{uv})$ denotes a specific set of links with respect to e_{uv} (see more details in [19]), and $\delta(e_{ij}) = \sum_{f_{pq}\in\mathcal{F}} d_{pq}x_{ij}^{pq}/b_{ij}$. In this way, the LP-relaxed problem $\widetilde{\mathcal{P}}_0$ is presented as follows:

$$\min \quad \mathcal{C}_{\mathcal{AB}} + \beta \times \mathcal{T}_{\mathcal{B}}$$

$$\text{subject to} \quad x_{ij}^{pq} \in [0,1], y_{ij} \in [0,1] \tag{15}$$

$$\delta(e_{ij}) + \sum_{e_{uv}\in\Omega(e_{ij})} \delta(e_{uv}) \leq 1, \ \forall e_{ij} \in E \tag{16}$$

$$\text{Constraints} \ \ (1),(2),(8),(9) \tag{17}$$

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 fractional solution to 0-1 integer solutions one by one.

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 $\overline{\mathbf{Y}} = \{y_{ij}\colon\ y_{ij} \in [0,1]\}$ 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.

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 $\sum_i w_i \leq 1$ [19]. 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.

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

Input: A conflict graph $G(V,E)$ and a weight vector $\{x(v) : v \in V\}$ where $x(v) \in [0,1]$
Output: The decomposed ISs $\{I\}$ with weights $\{w\}$

1: $\Gamma \leftarrow \emptyset$, $\Phi \leftarrow \{v \in V : x(v) > 0\}$
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**

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 detailed proof in our technical report [22]. ■

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 detailed proof in our technical report [22]. ■

Next, we fix the VM placement in $\mathcal{P}_0$ and solve the remaining LP problem. Then we decompose and round the fractional

[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 [25]. 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.

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 detailed proof in our technical report [22]. ■

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 [26] 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 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 [5] 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 [27] 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.

E. Other important generalized applications

Benefited from the high-level abstract nature of our PDR solution, we will show its important generalized applications.

Application of OCSs and FSOs. First, we show how to extend our model to the case of using OCSs [11], [17] or FSOs [9], [12] in DCNs. The OCS changes the topology by solving a bipartite matching from its input ports to the output ports [17]. 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 [23]. 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 [12]. Hence it can also be modeled as the independent set and the PDR solves it with the same approximation ratio.

Application of SINR model. Second, 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 [24] 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 $O(\log^4 n / \log \log^3 n)$.

IV. Evaluation

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

A. Simulation setup

For simulations, we use the public traffic trace from two university data centers provided by [28]. We implement a flow-level simulator using the TCP setting in [29]. We use a 3-layer *fat-tree* [30] with 8-port switches as the simulated network topology. To embed realistic wireless settings, we use the 60GHz rectangular waveguide hardware of AINFO Incs 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 [14], 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 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 [8], [31]; (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 [5], [18]. To compare with our algorithm, without loss of generality, we use the greedy-fill algorithm (named *Greedy*) in [31] and the clustering-and-matching (named *Cluster*) algorithm in [18] 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 basicline for VM placement.

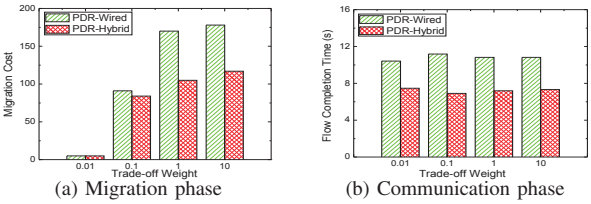
For the wireless setup, we compare our wireless solution with the greedy scheme (named *Flyway*) proposed in [15], which builds wireless links between close-by racks that have high capacity. For the routing strategy, we use the most popular DCN routing scheme ECMP (Equal-Cost Multi-Path) [32] to serve 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.

B. Simulation results

1) Trade-off performance: As Fig. 2 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. 2a, 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 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. 2(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.

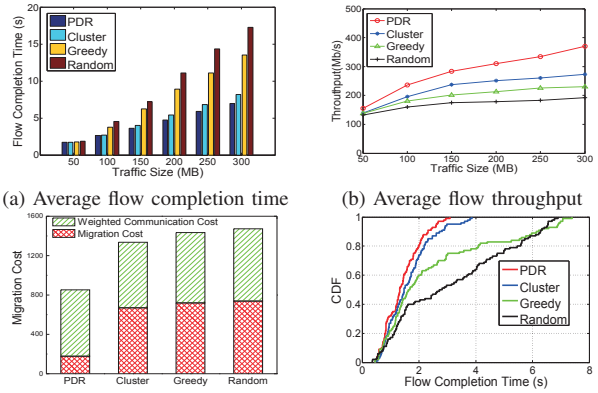
TABLE II. Algorithm Comparison on Different Aspects

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



(a) Migration phase (b) Communication phase

Fig. 2. Performance over different trade-off weights

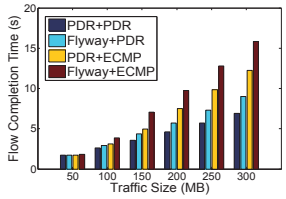


(a) Average flow completion time (b) Average flow throughput

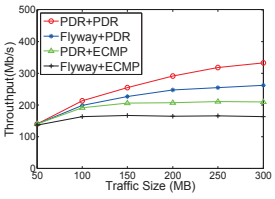
(c) Total cost on VM migration and communication (d) CDF of flow completion time

Fig. 3. Performance of different VM placement strategies

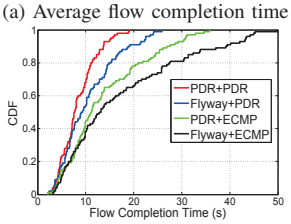
2) Performance of VM placement: In Fig. 3, we evaluate four different VM placement solutions. To make a fair comparison, all the compared solutions use exactly the same wireless setup as PDR, and also compute their routing paths by solving the remaining flow routing LP problem as that done by PDR. In Fig. 3a and Fig. 3b, 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



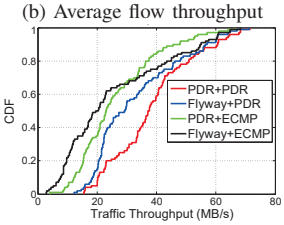
(a) Average flow completion time



(b) Average flow throughput



(c) CDF of flow completion time



(d) CDF of flow throughput

Fig. 4. Flow performance of different wireless setup and routing strategies

that the greedy placement algorithm works more effectively in a non-congested network.

The Fig. 3c and Fig. 3d show the detailed performance. In Fig. 3c, 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. 3d 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.

3) Performance of wireless setup and routing: In Fig. 4, 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. 4a and Fig. 4b, 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. 4c and Fig. 4d show that the maximum completion time of 90% flows in PDR is about 30% less than that of other solutions. Moreover, in PDR

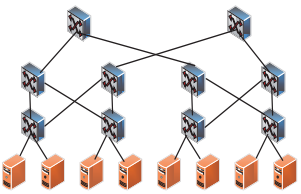


Fig. 5. Topology of testbed

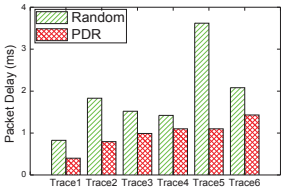


Fig. 6. Average packet delay

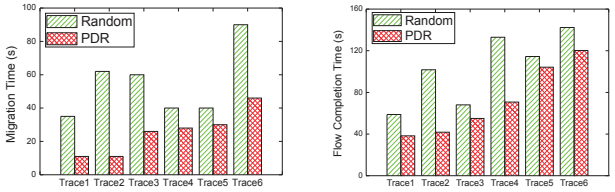
there are only 20% *slow flows* that have throughputs less than 25MBps, while there are at least 40% such flows in the others.

C. Experiment results

In this section, we present a hardware testbed to validate the performance gains of our PDR solution. 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* [33]. We use *VirtualBox v4.1.12* [34] to implement the virtual servers and use *OpenvSwitch v1.4.6* [35] 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. 5).

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 [14]. 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. 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.

As Fig. 6 shows, for all the traces, the flows in PDR have a much lower average packet delay than Random, and the difference is about 48.5% on the average. This is because the 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 flow performance in Fig. 7. In Fig. 7a, we can see that the total migration time of PDR is smaller than that of Random (improved by about 53.5% on average and up to only one sixth of that of Random). Besides this low migration cost,



(a) Total migration time (b) Flow completion time

Fig. 7. Flow performance in the testbed

as Fig. 7b shows, for all the traces, the flows in PDR still achieve a smaller completion time compared to that of Random (improved by about 30.4% on average). This demonstrates the effectiveness of our PDR solution to ensure the high performance of all the flows in a real testbed.

V. 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 breakthrough 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. Then we develop a powerful PDR algorithm to solve the optimization problem in polynomial time with a proved approximation ratio. Further, we show that our PDR solution can be extended to solve the VM migration optimization under different topology adaption technologies with proved approximation ratios. The real-trace based simulations demonstrate the advantage of our solution in achieving a smaller flow completion time while at same time ensuring a much lower cost for VM migration compared to other state-of-art VM migration solutions. Finally, we validate the feasibility of our solution to improve the flow performance in a testbed.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (no. 61422206, 61120106008), Tsinghua National Laboratory for Information Science and Technology (TNList). Xin Wang's research is supported by NSF CNS 1526843.

References

[1] V. Medina and J. M. García, "A survey of migration mechanisms of virtual machines," *ACM Computing Surveys (CSUR)*, vol. 46, no. 3, p. 30, 2014.
[2] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *NSDI 2005*.
[3] J. Zhang, F. Ren, and C. Lin, "Delay guaranteed live migration of virtual machines," in *INFOCOM 2014*.
[4] H. Wang, Y. Li, Y. Zhang, and D. Jin, "Virtual machine migration planning in software-defined networks," in *INFOCOM 2015*.
[5] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *INFOCOM 2010*.
[6] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint vm placement and routing for data center traffic engineering," in *INFOCOM 2012*.
[7] 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 Performance Evaluation Review*, vol. 41, no. 3, pp. 107–112, 2014.
[8] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *INFOCOM 2011*.
[9] M. Ghobadi, R. Mahajan, A. Phanishayee, H. Rastegarfar, P.-A. Blanche, M. Glick, D. Kilper, J. Kulkarni, G. Ranade, and N. Devanur, "Projector: Agile reconfigurable datacenter interconnect," in *SIGCOMM 2016*.
[10] Y. Cui, S. Xiao, X. Wang, Z. Yang, C. Zhu, X. Li, L. Yang, and N. Ge, "Diamond: Nesting the data center network with wireless rings in 3d space," in *NSDI 2016*.
[11] H. Liu, M. K. Mukerjee, C. Li, N. Feltman, G. Papen, S. Savage, S. Seshan, G. M. Voelker, D. G. Andersen *et al.*, "Scheduling techniques for hybrid circuit/packet networks," in *CoNEXT 2015*.
[12] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer, "Firefly: a reconfigurable wireless data center fabric using free-space optics," in *SIGCOMM 2014*.
[13] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating microsecond circuit switching into the data center," in *SIGCOMM 2013*.
[14] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng, "Mirror mirror on the ceiling: flexible wireless links for data centers," in *SIGCOMM 2012*.
[15] D. Halperin, S. Kandula, J. Padhye, P. Bahl, and D. Wetherall, "Augmenting data center networks with multi-gigabit wireless links," in *SIGCOMM 2011*.
[16] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, and M. Ryan, "c-through: Part-time optics in data centers," in *SIGCOMM 2010*.
[17] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: a hybrid electrical/optical switch architecture for modular data centers," in *SIGCOMM 2010*.
[18] 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," *Computer Networks*, vol. 57, no. 1, pp. 179–196, 2013.
[19] P.-J. Wan, "Multiflows in multihop wireless networks," in *MobiHoc 2009*.
[20] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide to the theory of np-completeness," *WH Freeman & Co., San Francisco*, pp. 61–62, 1979.
[21] S. Sen, D. Shue, S. Ihm, and M. J. Freedman, "Scalable, optimal flow routing in datacenters via local link balancing," in *CoNEXT 2013*.
[22] S. Xiao, Y. Cui, X. Wang, Z. Yang, L. Yang, and S. Yan, "Traffic-aware virtual machine migration in topology-adaptive dcn," Tsinghua University, Tech. Rep., 2016. [Online]. Available: http://www.4over6.edu.cn/others/WiVM.pdf
[23] "Matching polytope," http://www.imsc.res.in/∼meena/matching/polytope.
[24] M. Hoefer, T. Kesselheim, and B. Vöcking, "Approximation algorithms for secondary spectrum auctions," *ACM Transactions on Internet Technology (TOIT)*, vol. 14, no. 2-3, p. 16, 2014.
[25] R. Carr and S. Vempala, "Randomized metarounding," in *STOC 2000*.
[26] 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.
[27] D. Richert and J. Cortés, "Robust distributed linear programming," *Automatic Control, IEEE Transactions on*, vol. 60, no. 10, pp. 2567–2582, 2015.
[28] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *IMC 2010*.
[29] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks." in *NSDI 2010*.
[30] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *SIGCOMM 2008*.
[31] D. Erickson, B. Heller, N. McKeown, and M. Rosenblum, "Using network knowledge to improve workload performance in virtualized data centers," in *IEEE IC2E 2014*.
[32] C. E. Hopps, "Analysis of an equal-cost multi-path algorithm," *RFC 2992, IETF*, 2000.
[33] "Floodlight sdn controller," http://www.projectfloodlight.org/floodlight/.
[34] "Virtual box," https://www.virtualbox.org/.
[35] "Open vswitch," http://openvswitch.org/.