Virtual Machine Migration Planning in Software-Defined Networks

Huandong Wang, Yong Li, *Member, IEEE,* Ying Zhang, *Member, IEEE,* Depeng Jin, *Member, IEEE,*

Abstract—Live migration is a key technique for virtual machine (VM) management in data center networks, which enables flexibility in resource optimization, fault tolerance, and load balancing. Despite its usefulness, the live migration still introduces performance degradations during the migration process. Thus, there has been continuous efforts in reducing the migration time in order to minimize the impact. From the network's perspective, the migration time is determined by the amount of data to be migrated and the available bandwidth used for such transfer. In this paper, we examine the problem of how to schedule the migrations and how to allocate network resources for migration when multiple VMs need to be migrated at the same time. We consider the problem in the Software-defined Network (SDN) context since it provides flexible control on routing.

More specifically, we propose a method that computes the optimal migration sequence and network bandwidth used for each migration. We formulate this problem as a mixed integer programming, which is NP-hard. To make it computationally feasible for large scale data centers, we propose an approximation scheme via linear approximation plus fully polynomial time approximation, and obtain its theoretical performance bound and computational complexity. Through extensive simulations, we demonstrate that our fully polynomial time approximation (FPTA) algorithm has a good performance compared with the optimal solution of the primary programming problem and two state-of-the-art algorithms. That is, our proposed FPTA algorithm approaches to the optimal solution of the primary programming problem with less than 10% variation and much less computation time. Meanwhile, it reduces the total migration time and service downtime by up to 40% and 20% compared with the state-of-the-art algorithms, respectively.

Index Terms—VM migration planning, Software-defined Network (SDN), Migration sequence, Minimum migration time.

1 Introduction

As the increasing number of smart devices and various applications, IT services have been playing an important role in our daily life. In recent years, the quality and resilience of services have been drawing increasing attention. For example, it is important to optimize the delay, jitter and packet less rate of the service [2], as well as evacuate it before damages caused by disasters like earthquake to a suitable safe location [3]. On the other hand, since the advantage of the virtualization technology, it has been increasingly adopted in modern data centers and related infrastructures. Separating the software from the underlying hardware, virtual machines (VMs) are used to host various cloud services [1]. VMs can share a common physical host as well as be migrated from one host to another. Live migration, $i.e.$, moving VMs from one physical machine to another without disrupting services, is the fundamental technique that enables flexible and dynamic resource management in the virtualized data centers. Through applying these technologies, it is possible to optimize various parameters of IT services such as delay, packet loss rate by adjusting the locations of VMs dynamically, as well as evacuate services quickly before disasters by migrating the VMs from the dangerous site to a suitable location.

While there are continuous efforts on the optimal VM placements to reduce network traffic [5], [6], VM migration has received relatively less attention. We argue that careful planning of VM migration is needed to improve the system performance. First of all, in the case of evacuating services before disasters or recovering services after damages, we must migrate all VMs under constraints such as limited time or improve the impaired quality of services in time. Thus, it is important for us to minimize the evacuation time, $i.e.$, the total migration time. Then, the migration process consumes not only CPU and memory resources at the source and the migrated target's physical machines [6], [7], but also the network bandwidth on the path from the source to the destination [6]. The amount of available network resource has a big impact on the total migration time, $e.g.$, it takes longer time to transfer the same size of VM image with less bandwidth. As a consequence, the prolonged migration time should influence the application performance. Moreover, when multiple VM migrations occur at the same time, we need an intelligent scheduler to determine which migration tasks to occur first or which ones can be done simultaneously, in order to minimize the total migration time.

More specifically, there can be complex interactions between different migration tasks. While some independent migrations can be performed in parallel, other migrations may share the same bottleneck link in their paths. In this case, performing them simultaneously leads to longer total migration time. In a big data center, hundreds of migration requests can take place in a few minutes [8], where the effect of the migration order becomes more significant. Therefore, we aim to design a migration plan to minimize the total migration time by determining the orders of multiple migration tasks, the paths taken by each task, and the transmission rate of each task.

H. Wang, Y. Li and D. Jin are with State Key Laboratory on Microwave and Digital Communications, Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (liyong07@tsinghua.edu.cn, jindp@tsinghua.edu.cn).

Y. Zhang is with Ericsson Research Silicon Valley Lab, San Jose, CA, USA.

There have been a number of works on VM migration in the literature. Work [11], [35] focused on minimizing migration cost by determining an optimal sequence of migration. However, their algorithms were designed under the model of one-by-one migration, and thus cannot perform migration in parallel simultaneously, leading to a bad performance in terms of the total migration time. Bari *et al*. [9] also proposed a migration plan of optimizing the total migration time by determining the migration order. However, they assumed that the migration traffic of one VM only can be routed along one path in their plan. Compared with single-path routing, multipath routing is more flexible and can provide more residual bandwidth. Thus, we allow multiple VMs to be migrated simultaneously via multiple routing paths in our migration plan.

In this paper, we investigate the problem of how to reduce the total migration time in Software Defined Network (SDN) scenarios [10], [11]. We focus on SDN because with a centralized controller, it is easier to obtain the global view of the network, such as the topology, bandwidth utilization on each path, and other performance statistics. On the other hand, SDN provides a flexible way to install forwarding rules so that we can provide multipath forwarding between the migration source and destination. In SDN, the forwarding rules can be installed dynamically and we can split the traffic on any path arbitrarily. We allow multiple VMs to be migrated simultaneously via multiple routing paths. The objective of this paper is to develop a scheme that is able to optimize the total migration time by determining their migration orders and transmission rates. Our contribution is threefold, and is summarized as follows:

- We formulate the problem of VM migration from the network's perspective, which aims to reduce the total migration time by maximizing effective transmission rate in the network, which is much easier to solve than directly minimizing the total migration time. Specifically, we formulate it as a mixed integer programming (MIP) problem, which is NP-hard.
- We propose an approximation scheme via linear approximation plus fully polynomial time approximation, termed as FPTA algorithm, to solve the formulated problem in a scalable way. Moreover, we obtain its theoretical performance bound and computational complexity.
- By extensive simulations, we demonstrate that our proposed FPTA algorithm achieves good performance in terms of reducing total migration time, which reduces the total migration time by up to 40% and shortens the service downtime by up to 20% compared with the state-of-the-art algorithms.

The rest of the paper is organized as follows. In Section 2, we give a high-level overview of our system, and formulate the problem of maximizing effective transmission rate in the network. In Section 3, we propose an approximation scheme composed of a linear approximation and a fully polynomial time approximation to solve the problem. Further, we provide their bounds and computational complexities. In Section 4, we evaluate the performance of our solution through extensive simulations. After presenting related works in Section 5, we draw our conclusion in Section 6.

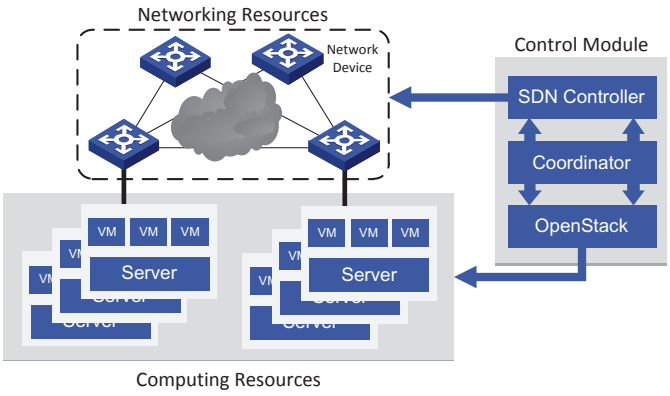


Fig. 1. System Overview

2 System Model and Problem Formulation

2.1 System Overview

We first provide a high-level system overview in this section. As shown in Fig. 1, in such a network, all networking resources are under the control of the SDN controller, while all computing resources are under the control of some cloud management system, such as OpenStack. Our VM migration plan runs at the Coordinator and it is carried out via the OpenStack and SDN controller.

More specifically, devices in the network, switches or routers, implement forwarding according to their obtained forwarding tables and do some traffic measurement. The SDN controller uses a standardized protocol, OpenFlow, to communicate with these network devices, and gathers link-state information measured by them. Meanwhile, the SDN controller is responsible for computing the forwarding tables for all devices. On the other hand, the cloud controller, OpenStack, is responsible for managing all computing and storage resources. It keeps all the necessary information about virtual machines and physical hosts, such as the memory size of the virtual machine, the residual CPU resource of the physical host. Meanwhile, all computing nodes periodically report their up-to-date information to it. Besides, OpenStack also provides general resource management functions such as placing virtual machines, allocating storage, etc. On the other hand, the OpenStack component, Neutron, manages the virtual networking, e.g., the communication between VMs, which is realized by collaborating with the SDN controller. The states of the virtual network, which can be obtained from Neutron, can be also used to implement applications such as optimizing the delay of traffic between VMs.

The processes of VM migration are described as follows. Firstly, migration requests of applications are sent to the Coordinator. Based on the data collected from the OpenStack and SDN controller, our proposed VM migration plan outputs a sequence of the VMs to be migrated with their corresponding bandwidths. Then, these migration requests are sent to the OpenStack from the Coordinator with the determined orders. After accepting a migration request, the OpenStack first communicates with the SDN controller to build a path from the source host to the target host through its networking component, Neutron. Specifically, they finish

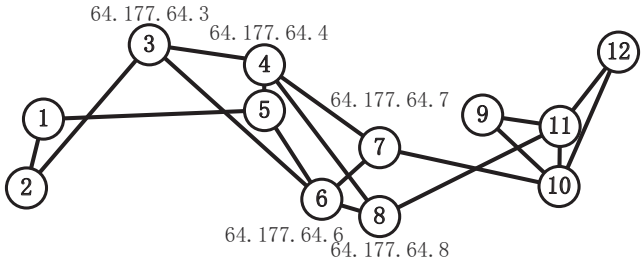
the reconfigure the flow tables of switches on the migration path, and provide a bandwidth guarantee. Then, a linkage is established from the source host to the target host, and the VM migration plan is carried out at the corresponding time by OpenStack. During migrations, the Coordinator is still monitoring the network states and process of migrations, and dynamically adjusts the migration sequence of VMs as well as their corresponding bandwidths in tune with the changing network conditions.

On the other hand, live migration is designed to move VMs from one physical machine to another with the minimal service downtime. It uses a pre-copy phase to transmit the memory of the migrating VM to the destination physical machine while it is still running to minimize the size of the data that should be transmitted in the downtime. However, pages that are modified during the pre-copy phase must be iteratively re-sent to ensure memory consistency. We use the page dirty rate to represent the size of memory that is modified per second during the pre-copy phase. Thus, the migration time of each VM depends on the memory size, the page dirty rate, and the transmission rate of it.

To compute the migration sequence of VMs, we need network topology and traffic matrix of the data center. Besides, memory sizes and page dirty rates of VMs, and residual physical resources such as CPU and memory are also needed. Most of them can be obtained directly from the SDN controller or OpenStack, but measurements of page dirty rate and traffic matrix need special functions of the platform. We next present the approach to measure them in details:

Page Dirty Rate Measurement: We utilize a mechanism called shadow page tables provided by Xen [1] to track dirtying statistics on all pages [2]. All page-table entries (PTEs) are initially read-only mappings in the shadow tables. Modifying a page of memory would result a page fault and then it is trapped by Xen. If write access is permitted, appropriate bit in the VMs dirty bitmap is set to 1. Then by counting the dirty pages in an appropriate period of time, we obtain the page dirty rate.

Traffic Measurement: We assume SDN elements, switches and routers, can spilt traffic on multiple next hops correctly, and perform traffic measurements at the same time [13], [14]. To aid traffic measurements, an extra column in the forwarding table is used to record the node in the network that can reach the destination IP address as in [13]. Take Fig. 2(a), which is the topology of the inter-datacenter WAN of google, as an example, where all nodes are SDN forwarding elements. For instance, we assume node 8 (IP address 64.177.64.8) is the node that can reach the subset 195.112/16, and the shortest path from node 7 to node 8 goes through node 6 (IP address 64.177.64.6). Then, the forwarding table of node 7 is shown in Fig. 2(b), where the first entry is corresponding to the longest matched prefix 195.112/16. When a packet with the longest matched prefix 195.112/16 is processed by node 7, α showed in Fig. 2(b) increases by the packet length. Thus, it tracks the number of bytes routed from node 7 to node 8 with the longest matched prefix 195.112/16. Using these data, the SDN controller can obtain the traffic matrix of the network, for example, by querying the SDN elements [15]. Tootoonchian *et al.* [15] proposed a traffic matrix estimation system for



(a) Google's inter-datacenter WAN

Prefix	Node	Next Hop	Traffic
195.112/16	64.177.64.8	64.177.64.6	α
195.027/16	64.177.64.3	64.177.64.4	β
...	...	...	...

(b) Modified Forwarding Table

Fig. 2. Google's inter-datacenter WAN and the modified forwarding table for example.

OpenFlow networks called OpenTM, which intelligently chooses switches to query and combines the measurements to estimate the traffic matrix. At the same time, it does not impose much overhead on the network.

2.2 Problem Overview

In this section, we provide an overview of our problem and then give an example of optimizing the total migration time by determining the migration orders and transmission rates of VMs.

In a cloud data center or inter-datacenter WAN, VMs hosted on physical machines are leveraged to provide various services. By virtualization, software is separated from its underlying hardware, and thus VMs can share a common physical host as well as be migrated from one host to another. In order to provide better services or troubleshoot when some physical infrastructure failures happen, data centers adjust the locations of VMs, demanding us to migrate series of VMs from their source hosts to target hosts. We assume there is no alternate network dedicated to VM migrations, because of the cost of its deployment, especially in large-scale infrastructures. Thus, only residual bandwidth can be used to migrate VMs. Then, our goal is to determining the VMs' migration orders and transmission rates that satisfy various constraints, such as capacity constraints for memory and links, to optimize the total migration time.

Now we give an example in Fig. 3. In this network, there are 2 switches (S_1 and S_2) and 4 physical machines (H_1 to H_4) hosting 3 VMs (V_1 to V_3). Assume the capacity of each link is 100MBps and memory size of each VM is 500MB. Then, there is a host failure happening at H_3. Thus, we must move all VMs on H_3 to other locations. To satisfy various constraints such as limitation of storage, we finally decide to migrate V_1 to H_1, V_2 to H_2, and V_3 to H_4, respectively. The optimal plan of migration orders and transmission rates is that first migrate V_1 and V_3 simultaneously, respectively with paths $\{(H_3, S_1, H_1)\}$ and $\{(H_3,S_2,H_4)\}$ and the corresponding maximum bandwidths

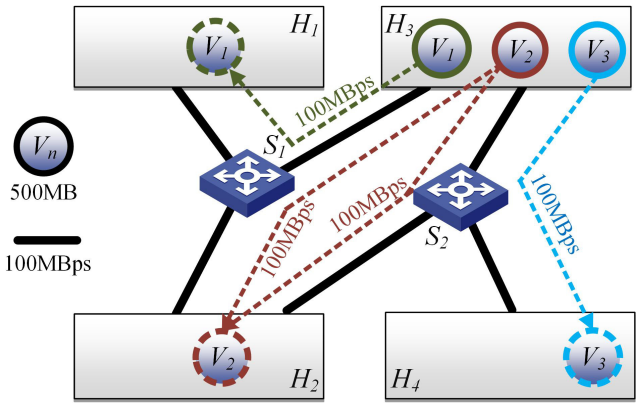


Fig. 3. An example of migration request and plan.

of 100MBps. Then migrate V_2 with paths $\{(H_3, S_1, H_2), (H_3, S_2, H_2)\}$ and the corresponding maximum bandwidth of 200MBps. It takes 7.5s in total to finish all the migrations. Then, take random migration orders for example, *i.e.*, first migrate V_1 and V_2 simultaneously, respectively with paths $\{(H_3, S_1, H_1)\}$ and $\{(H_3, S_2, H_2)\}$ and the corresponding maximum bandwidths of 100MBps. Then migrate V_3 with path $\{(H_3, S_2, H_4)\}$ and the corresponding maximum bandwidth of 100MBps. It takes 10s in total to finish all the migrations.

In this example, V_1 and V_3 can be migrated in parallel, while V_2 can be migrated with multipath. However, V_1 and V_2, V_3 and V_2 share same links in their paths, respectively. By determining a proper order, these migrations can be implemented making full use of the network resources. Thus, the total migration time is reduced by 25% in the example, illustrating the effect of the migration plan.

2.3 Mathematical Model for Live Migration

In this section, we present the mathematical model of live migration. Since the pre-copy migration is the most common and widely-used approach, and currently cannot be replaced by other methods, we mainly focus on the planning based on pre-copy migration in our work. We use M to represent the memory size of the virtual machine. Let R denote the page dirty rate during the migration and L denote the bandwidth allocated for the migration. Then, the process of the live migration is shown in Fig. 4. As we can observe, live migration copies memory in several rounds. Assume it proceeds in n rounds, and the data volume transmitted at each round is denoted by V_i $(0 \leq i \leq n)$. At the first round, all memory pages are copied to the target host, and then in each round, pages that have been modified in the previous round are copied to the target host. Thus, the data transmitted in round i can be calculated as:

$$V_i = \begin{cases} M & \text{if } i = 0, \\ R \cdot T_{i-1} & \text{otherwise.} \end{cases} \tag{1}$$

Meanwhile, the elapsed time at each round can be calculated as:

$$T_i = \frac{R \cdot T_{i-1}}{L} = \frac{M \cdot R^i}{L^{i+1}}, \tag{2}$$

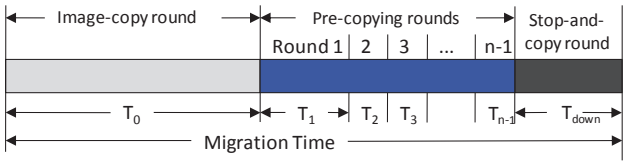


Fig. 4. Illustration of live migration performing pre-copy in iterative rounds.

where R^i and L^i is the i-th power of R and L, respectively. Assume the page dirty rate is smaller than the bandwidth allocated for the migration. Let λ denote the ratio of R to L, that is:

$$\lambda = R/L. \tag{3}$$

If $R \geq L$, then for all $i \geq 1$ we have $V_i = R \cdot T_{i-1} = R \cdot V_{i-1}/L = \lambda \cdot V_{i-1} \geq V_{i-1}$. It means that the remaining dirty memory does not reduce after each round, and we will never finish the migration. Thus, we assume $R < L$, and we have $\lambda < 1$.

Combining the above analysis, the total migration time can be represented as:

$$T_{mig} = \sum_{i=0}^{n} T_i = \frac{M}{L} \cdot \frac{1 - \lambda^{n+1}}{1 - \lambda}. \tag{4}$$

Let V_{thd} denote the threshold value of the remaining dirty memory that should be transferred at the last iteration. We can calculate the total rounds of the iteration by the inequality $V_n \leq V_{thd}$. Using the previous equations we obtain:

$$n = \left\lceil \log_\lambda \frac{V_{thd}}{M} \right\rceil. \tag{5}$$

In this model, the downtime caused in the migration can be represented as:

$$T_{down} = T_d + T_r, \tag{6}$$

where T_d is the time spent on transferring the remaining dirty pages, and T_r is the time spent on resuming the VM at the target host. For simplicity, we assume the size of remaining dirty pages is equal to V_{thd}.

2.4 Problem Formulation

The network is represented by a graph $G = (V, E)$, where V denotes the set of network nodes and E denotes the set of links. Let $c(e)$ denote the residual capacity of the link $e \in \mathbf{E}$. Let a migration tuple (s_k, d_k, m_k, r_k) denote that a virtual machine should be migrated from the node s_k to the node d_k with the memory size m_k and the page dirty rate r_k. There are totally K migration tuples in the system. For the migration k, l_k represents the bandwidth allocated for it. Let $\mathbf{P}_k$ denote the set of paths between s_k and d_k. The flow in path p is represented by the variable $x(p)$. Besides, as different migrations are started at different times, we define binary variable X_k to indicate whether migration k has been started at the current time.

We first discuss the optimization objective. To obtain an expression of the total migration time is difficult in our

TABLE 1
List of commonly used notations.

Notation	*Description*
M	The memory size of the virtual machine.
R	The page dirty rate during the migration.
L	The bandwidth allocated for the migration.
V_i	The data transmitted in round i.
T_i	The ratio of R to L.
λ	The memory size of the virtual machine.
T_{mig}	The total migration time.
T_{down}	The downtime caused in the migration.
$\mathbf{V}, n$	Set of network nodes and the number of network nodes.
$\mathbf{E}, m$	Set of network links and the number of network links.
K	The number of VMs to be migrated.
H	The number of hosts in the network.
$c(e)$	Residual capacity of the link $e \in \mathbf{E}$.
s_k	Source host of the migration k.
d_k	Target host of the migration k.
m_k	Memory size of the migration k.
l_k	Bandwidth allocated for the migration k.
r_k	Page dirty rate of the virtual machine corresponding to the migration k.
X_k	Binary variable indicating whether migration k has been started.
$\mathbf{P}$	Set of all paths in the network.
$\mathbf{P}_k$	Set of paths between s_k and d_k.
$\mathbf{P}_e$	Set of paths using link $e \in \mathbf{E}$.
$x(p)$	The amount of flow in path $p \in \mathbf{P}$.
l^*, N^*	The optimal solution for (10) with fewest nonzero variables and the number of nonzero variables in it.
$\hat{l}, \hat{N}$	The optimal solution for (12) such that at least N^* equalities that hold in inequality constraints of (12) and the number of nonzero variables in it.
$u(e)$	The dual variables of the problem (11).
dist(p)	$\sum_{e \in p} u(e)$, the length of path p in the dual problem of (11).
U	The optimal value of the MIP problem (8).
V	The optimal value of the LP problem (12).
W	The transmission rate corresponding to the solution of the FPTAS algorithm.
F	The net transmission rate corresponding to the solution of the FPTA algorithm.
σ	Accuracy level of the linear approximation.
ϵ	Accuracy level of the FPTAS algorithm.
κ	Accuracy level of the FPTA algorithm.

model, because we allow multiple VMs to be migrated simultaneously. Thus, the total migration time cannot simply be represented as the sum of the migration time of each VM like [11], [35], whose migration plans were designed under the model of one-by-one migration. Moreover, even though we obtain the expression of the total migration time, the optimization problem is still difficult and cannot be solved efficiently. For example, work [9] gives an expression of the total migration time by adopting a discrete time model. However, they did not solve the problem directly, instead, they proposed a heuristic algorithm independent with the formulation without any theoretical bound. Thus, we try to obtain the objective function reflecting the total migration time from other perspectives.

On the other hand, since the downtime of live migration is required to be unnoticeable by users, the number of the remaining dirty pages in the stop-and-copy round, *i.e.* V_n, need to be small enough. According to the model provided in the last subsection, we have $V_n = M \cdot \lambda^n$. Thus, λ^n must be small enough. For example, if migrating a VM, whose memory size is 10GB, with the transmission rate of 1GBps, to reduce the downtime to 100ms, we must ensure $\lambda^n \leq 0.01$. Thus, by ignoring λ^n in the equation (4), we have:

$$T_{mig} \approx \frac{M}{L} \cdot \frac{1}{1-\lambda} = \frac{M}{L-R}. \tag{7}$$

We call the denominator as $net\ transmission\ rate$. From an overall viewpoint, the sum of memory sizes of VMs is reduced with the speed of $\sum_{k=1}^{K}(l_k - X_k r_k)$, which is the total net transmission rate in the network. In turn, the integration of the net transmission rate respect to time is the sum of memory sizes. Note that this conclusion also works when the transmission rate is smaller than the page dirty rate, as well as when the transmission rate changes with time. In this case, we also need to maximize the total net transmission rate to reduce the remaining memory to be transmitted. By maximizing the total net transmission rate, we can reduce the total migration time efficiently. Thus, it is reasonable for us to convert the problem of reducing the migration time to maximizing the net transmission rate, which is expressed as $\sum_{k=1}^{K}(l_k - X_k r_k)$.

We now analyze constraints of the problem. A VM is allowed to be migrated with multipath in our model. Thus, we have a relationship between l_k and $x(p)$:

$$\sum_{p \in \mathbf{P}_k} x(p) = l_k, \quad k = 1, ..., K.$$

Note that $x(p)$ in equation is dependent on k. However, since the existence of page dirty rate, if there are multiple migration requests with the same source and target hosts, in order to minimize the total migration time, it is better to migrate them one by one rather than migrate them simultaneously. Thus, given a group of VM migration requests, we each time only consider one migration for each unique pair of source and target hosts. In this case, there can be only traffic of one migration on each path at one time. Thus, the dependence between $x(p)$ and k can be ignored.

Besides, the total flow along each link must not exceed its capacity. Thus, we have:

$$\sum_{p \in \mathbf{P}_e} x(p) \leq c(e), \ \ \forall e \in \mathbf{E}.$$

For a migration that has not been started, there is no bandwidth allocated for it. Thus, we have constraints expressed as follow:

$$l_k \leq \beta \cdot X_k, \quad k = 1, ..., K,$$

where β is a constant large enough so that the maximum feasible bandwidth allocated for each migration cannot exceed it. Then, the problem of maximizing the net transmission rate can be formulated as follows:

$$\begin{aligned} \textbf{max} \ & \textstyle\sum_{k=1}^{K}(l_k - X_k r_k) \\ \textbf{s.t.} \ & \begin{cases} \sum_{p \in \mathbf{P}_k} x(p) = l_k, & k = 1, ..., K \\ \sum_{p \in \mathbf{P}_e} x(p) \leq c(e), & \forall e \in \mathbf{E} \\ l_k \leq \beta \cdot X_k, & k = 1, ..., K \\ X_k \in \{0, 1\}, & k = 1, ..., K \\ x(p) \geq 0, & p \in \mathbf{P} \end{cases} \end{aligned} \tag{8}$$

which is a mixed integer programming (MIP) problem.

When some old migrations are finished, the input of the problem changes. Thus, we recalculate the programming

under the new updated input. We notice that migrations that have been started cannot be stopped. Otherwise, these migrations must be back to square one because of the effect of the page dirty rate. Thus, when computing this problem next time, we add the following two constraints to it:

$$\begin{cases} X_k \geq X_k^0, & k = 1, ..., K \\ l_k \geq l_k^0, & k = 1, ..., K \end{cases} \tag{9}$$

where X_k^0 and l_k^0 are equal to the value of X_k and l_k in the last computing, respectively. It means a migration cannot be stopped and its bandwidth does not decrease.

Remark that the network states, including the background traffic, page dirty rate of VMs, etc., are also changing all over the time. With the change of the input for the problem, the migration planning also need to be recalculated. Thus, in our approach, the programming will be recalculated with updated network state information periodically. On the other hand, network fluctuations will lead to significant change of network states. Thus, it will also trigger the re-planning of VM migrations. At certain time points, the bandwidth might be not enough, even smaller than the page dirty rate. In that case, our basic strategy, i.e., maximizing the total net transmission rate, still achieves the local optimum, but performance of our approximation algorithm may decrease. However, this performance gap can be evaluated and bounded by Theorem 2. If the performance gap is too large, we can dynamically adjust our strategies, e.g., temporarily give up the approximation algorithm and return to solve the primary problem as an alternative solution.

Another problem should be considered is the priorities of different migrations. In fact, migrations with shorter path tend to be carried out early. The reason is that they consume less resources compared with migrations with longer paths, and have less conflict with other migrations. Thus, by maximizing the total net transmission rate, they are more likely to be selected. However, in real scenarios, sometimes we may need to first complete migrations with long paths, such as migrations across datacenters. One solution is to support the different priorities between migrations in the scheduling process, which can be realised by replacing the total net transmission rate in the objective function by the weighted summation of the net transmission rates of different VMs. By setting suitable weight, migrations with long paths can be carry out in reasonable time.

By solving the programming, we obtain the VMs that should be migrated with their corresponding transmission rates, maximizing the total net transmission rate under the current condition. By dynamically determining the VMs to be migrated in tune with changing traffic conditions and migration requests, we keep the total net transmission rate maximized, which is able to significantly reduce the total migration time.

3 Approximation Algorithm

Solving the formulated MIP problem, we obtain a well-designed sequence of the VMs to be migrated with their corresponding bandwidths. However, the MIP problem is NP-hard, and the time to find its solution is intolerable on large scale networks. For example, we implement the MIP problem using YALMIP – a language for formulating generic optimization problems [16], and utilize the GLPK to solve the formulation [17]. Then, finding the solution of a network with 12 nodes and 95 VMs to be migrated on a Quad-Core 3.2GHz machine takes at least an hour. Therefore, we need an approximation algorithm with much lower time complexity.

3.1 Approximation Scheme

3.1.1 Linear Approximation

Let us reconsider the formulated MIP problem (8). In this problem, only $X_k,\ k = 1, ..., K$, are integer variables. Besides, the coefficient of X_k in the objective function is r_k. In practical data center, r_k is usually much less than l_k, $i.e.$, the migration bandwidth of the VM. Thus, we ignore the part of $\sum_{k=1}^{K} X_k r_k$ in the objective function, and remove variables $X_k,\ k = 1, ..., K$. Then, we obtain a linear programming (LP) problem as follows:

$$\begin{aligned} \textbf{max}\ & \textstyle\sum_{k=1}^{K} l_k \\ \textbf{s.t.}\ & \begin{cases} \sum_{p \in \mathbf{P}_k} x(p) = l_k, & k = 1, ..., K \\ \sum_{p \in \mathbf{P}_e} x(p) \leq c(e), & \forall e \in \mathbf{E} \\ x(p) \geq 0, & p \in \mathbf{P} \end{cases} \end{aligned} \tag{10}$$

We select the optimal solution l^* for (10) with most variables that are equal to zero as our approximate solution. Then we let N^* denote the number of variables that are not zero in our approximate solution l^*, and the corresponding binary decision variables X_k are then set to be 1, while the other binary decision variables are set to be 0. Then the final approximate solution is denoted by (l_k^*, X_k^*).

As for the primary problem with the additional constraints shown in (9), by a series of linear transformations, the problem is converted to a LP problem with the same form as (10) except for a constant in the objective function, which can be ignored. Thus we obtain a linear approximation for the primary MIP problem.

3.1.2 Fully Polynomial Time Approximation

The exact solution of the LP problem (10) still cannot be found in polynomial time, which means unacceptable computation time for large scale networks. Thus, we further propose an algorithm to obtain the solution in polynomial time at the cost of accuracy.

Actually, ignoring the background of our problem and removing the intermediate variable l_k, we can express the LP problem (10) as:

$$\begin{aligned} \textbf{max}\ & \textstyle\sum_{p \in \mathbf{P}} x(p) \\ \textbf{s.t.}\ & \begin{cases} \sum_{p \in \mathbf{P}_e} x(p) \leq c(e), & \forall e \in \mathbf{E} \\ x(p) \geq 0, & p \in \mathbf{P} \end{cases} \end{aligned} \tag{11}$$

This is a maximum multicommodity flow problem, that is, finding a feasible solution for a multicommodity flow network that maximizes the total throughput.

Fleischer $et\ al.$ [19] proposed a Fully Polynomial-time Approximation Scheme (FPTAS) algorithm independent of the number of commodities K for the maximum multicommodity flow problem. It can obtain a feasible solution whose

objective function value is within $1+\epsilon$ factor of the optimal, and the computational complexity is at most a polynomial function of the network size and $1/\epsilon$.

Specifically, the FPTAS algorithm is a primal-dual algorithm. We denote $u(e)$ as the dual variables of this problem. For all $e \in \mathbf{E}$, we call $u(e)$ as the length of link e. Then, we define dist$(p) = \sum_{e \in p} u(e)$ as the length of path p. This algorithm starts with initializing $u(e)$ to be δ for all $e \in \mathbf{E}$ and $x(p)$ to be 0 for all $p \in \mathbf{P}$. δ is a function of the desired accuracy level ϵ, which is set to be $(1+\epsilon)/((1+\epsilon)n)^{1/\epsilon}$ in the algorithm. The algorithm proceeds in phases, each of which is composed of K iterations. In the r_{th} phase, as long as there is some $p \in \mathbf{P}_k$ for some k with dist(p) <min$\{\delta(1+\epsilon)^r, 1\}$, we augment flow along p with the capacity of the minimum capacity edge in the path. The minimum capacity is denoted by c. Then, for each edge e on p, we update $u(e)$ by $u(e) = u(e)(1 + \frac{\epsilon c}{c(e)})$. At the end of the r_{th} phase, we ensure every (s_j, d_j) pair is at least $\delta(1+\epsilon)^r$ or 1 apart. When the lengths of all paths belonging to $\mathbf{P}_k$ for all k are between 1 and $1+\epsilon$, we stop. Thus, the number of phases is at most $\lceil log_{1+\epsilon}\frac{1+\epsilon}{\delta} \rceil$. Then, according to theorem in [19], the flow obtained by scaling the final flow obtained in previous phases by $\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$ is feasible. We modified the FPTAS algorithm by adding some post-processes to obtain the feasible (l_k, X_k) and $x(p)$ to the primal MIP problem, and the modified algorithm is given in more detail in Algorithm 1. The computational complexity of the post-processes is only a linear function of the network size and the number of the VMs to be migrated. In addition, the computational complexity of the FPTAS algorithm is at most a polynomial function of the network size and $1/\epsilon$ [19]. Thus, the computational complexity of our approximation algorithm is also polynomial, and we obtain a fully polynomial time approximation (termed as FPTA) to the primal MIP problem (8). In next subsections, we will analyze its performance bound and computational complexity in details.

3.2 Bound Analysis

To demonstrate the effectiveness of our proposed algorithm, we now analyze the bound of it. We first analyze the bound of the linear approximation compared with the primary MIP problem (8), then analyze the bound of the FPTA algorithm compared with the linear approximation (10). With these two bounds, we finally obtain the bound of the FPTA algorithm showing in Algorithm 1 compared with the primary MIP problem (8).

3.2.1 Bound of the Linear Approximation

We discuss the bound of the linear approximation compared with the primary MIP problem in the data center network with full bisection bandwidth. A bisection of a network is a partition into two equally-sized sets of nodes. The sum of the capacities of links between the two partitions is called the bandwidth of the bisection. The bisection bandwidth of a network is the minimum such bandwidth along all possible bisections. Therefore, bisection bandwidth can be thought of as a measure of worst-case network capacity, and the corresponding partitions is called the most critical bisection cut. A network is said to have a full bisection bandwidth, if it can sustain the full network access bandwidth of every node across the most critical bisection while all nodes communicate simultaneously. Common topologies of data center networks, such as fat tree, usually provide full bisection bandwidth.

In the network with full bisection bandwidth, it is possible for an arbitrary host in the data center to communicate with any other host in the network at the full bandwidth of its local network interface [38]. Thus, we can ignore the routing details, and only guarantee the traffic at each host not exceeds the full bandwidth of its local network interface. Then, the LP problem (10) becomes:

$$\begin{aligned} \textbf{max } & \textstyle\sum_{k=1}^{K} l_k \\ \textbf{s.t. } & \begin{cases} \sum_{s_k=i} l_k \le C_i^s, & i = 1, ..., H \\ \sum_{d_k=i} l_k \le C_i^d, & i = 1, ..., H \\ l_k \ge 0, & k = 1, ..., K \end{cases} \end{aligned} \tag{12}$$

where C_i^s is the maximum amount of traffic that can be received at host i, while C_i^d is the maximum amount of traffic that can be sent at host i. Besides, there are H hosts in the data center. Then, we let L_0 be the minimum of C_i^s and C_i^d. That is, min$\{C_1^s, ..., C_H^s\} \ge L_0$ and min$\{C_1^d, ..., C_H^d\} \ge L_0$. Similarly, we let R_0 be the maximum of r_k. That is, max$\{r_1, ..., r_K\} \le R_0$.

We now provide some supplement knowledge about linear programming. For a linear programming with standard

Algorithm 1: FPTA Algorithm.

Input: network $G(V, E)$, link capacities $c(e)$ for $\forall e \in \mathbf{E}$, migration requests (s_j, d_j)
Output: Bandwidth l_k, binary decision variable X_k for each migration k, and the amount of flow $x(p)$ in path $p \in \mathbf{P}$.
Initialize:
  $u(e) \leftarrow \delta \ \forall e \in \mathbf{E}$
  $x(p) \leftarrow 0 \ \forall p \in \mathbf{P}$
for $r = 1$ to $\lceil log_{1+\epsilon}\frac{1+\epsilon}{\delta} \rceil$ **do**
  for $j = 1$ to K **do**
    $p \leftarrow$ shortest path in $\mathbf{P}_j$
    while $dist(p) < \min\{1, \delta(1+\epsilon)^r\}$ **do**
      $c \leftarrow \min_{e \in p} c(e)$
      $x(p) \leftarrow x(p) + c$
      $\forall e \in p, u(e) \leftarrow u(e)(1 + \frac{\epsilon c}{c(e)})$
      $p \leftarrow$ shortest path in $\mathbf{P}_j$
for each $p \in \boldsymbol{P}$ **do**
  $x(p) = x(p)/\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$
for $j = 1$ to K **do**
  $l_j = \sum_{p \in \mathbf{P}_j} x(p)$
  $X_j = 0$
  if $l_j \neq 0$ **then**
    $X_j = 1$
Return (l_k, X_k) and $x(p)$

form, which can be represented as:

$$\begin{aligned} &\textbf{max } b^T x \\ &\textbf{s.t. } \begin{cases} Ax = c \\ x \geq 0 \end{cases} \end{aligned} \tag{13}$$

where $x, b \in R^n$, $c \in R^m$, $A \in R^{m \times n}$ has full rank m, we have the following definitions and lemmas.

Definition 1 (Basic Solution) Given the set of m simultaneous linear equations in n unknowns of $Ax = c$ in (13), let B be any nonsingular $m \times m$ submatrix made up of columns of A. Then, if all $n - m$ components of x not associated with columns of B are set equal to zero, the solution to the resulting set of equations is said to be a basic solution to $Ax = c$ with respect to the basis B. The components of x associated with columns of B are called basic variables, that is, $Bx_B = c$ [20].

Definition 2 (Basic Feasible Solution) A vector x satisfying (13) is said to be feasible for these constraints. A feasible solution to the constraints (13) that is also basic is said to be a basic feasible solution [20].

Lemma 1 (Fundamental Theorem of LP) Given a linear program in standard form (13) where A is an $m \times n$ matrix of rank m. If there is a feasible solution, there is a basic feasible solution. If there is an optimal feasible solution, there is an optimal basic feasible solution [20].

These definitions and the lemma with its proof can be found in the textbook of linear programming [20]. Thus we ignore the proof for the lemma. With these preparations, we have the following lemma:

Lemma 2 There exists an optimal solution for (12), such that there are at least N^* equalities that hold in inequality constraints of (12).

Proof: Problem (12) can be represented in matrix form as:

$$\begin{aligned} &\textbf{max } b^T l \\ &\textbf{s.t. } \begin{cases} Al \leq c \\ l \geq 0 \end{cases} \end{aligned} \tag{14}$$

where $l = (l_1, l_2, ..., l_K)^T$, $b = (1, 1, ..., 1)^T \in R^K$, $c \in R^{2H}$, $A \in R^{2H \times K}$. Besides, A is composed of 0 and 1, and each column of A has and only has two elements of 1. Then this problem can be transformed to standard form represented as:

$$\begin{aligned} &\textbf{max } b^T l \\ &\textbf{s.t. } \begin{cases} [A\ I] \begin{bmatrix} l \\ s \end{bmatrix} = c \\ l, s \geq 0 \end{cases} \end{aligned} \tag{15}$$

where $s = c - Al \in R^{2H}$, $I \in R^{K \times K}$ is the identity matrix of the order K. Besides, $[A\ I] \in R^{2H \times (K+2H)}$ has full rank $2H$.

By Lemma 1, if the LP problem (15) has an optimal feasible solution, we can find an optimal basic feasible solution $(\hat{l}, \hat{s})$ for (15). Then $\hat{l}$ is an optimal solution for (14). By the definition of basic solution, the number of nonzero variables in $(\hat{l}, \hat{s})$ is less than $2H$. Meanwhile, By the definition of N^*, the number of nonzero variables in $\hat{l}$, which is represented by $\hat{N}$, is greater than N^*. Thus the number of nonzero variables in $\hat{s}$ is less than $2H - N^*$. Then there are at least N^* variables that are equal to zero in $\hat{s}$. Meanwhile, $\hat{s}_j = 0,\ j \in \{1, ..., 2H\}$ means the equality holds in the inequality constraint corresponding j_{th} row in A. Therefore, we have at least N^* equalities that hold in inequality constraints of (12).■

Theorem 1 Assume $R_0 = \eta L_0$. Let U be the optimal value of the primal MIP problem (8), and V be the optimal value of the LP problem (12). Then we have $V - N^* R_0 \geq (1 - \sigma)U$, where $\sigma = \frac{2\eta}{1-2\eta}$.

Proof: We first prove $V \geq \frac{1}{2} N^* L_0$. By lemma 2, we know that there exists an optimal solution of (12) such that there are at least N^* equalities that hold in inequality constraints of (12). We select the corresponding rows $a_1, ...a_{N^*}$ of A and corresponding elements $c_1, ...c_{N^*}$ of c. Then we have $a_i^T \hat{l} = c_i,\ i = 1, ..., N^*$. Because each column of A has and only has two elements of 1, elements of $\sum_{i=1}^{N^*} a_i$ are at most 2. Thus, we have $V = \sum_{k=1}^{K} \hat{l}_k \geq \frac{1}{2} \sum_{i=1}^{N^*} a_i^T \hat{l} = \frac{1}{2} \sum_{i=1}^{N^*} c_i \geq \frac{1}{2} N^* L_0$.

By definition of U and V, we have $U \leq V$ and $V - N^* R_0 \leq U$. Then we have $|U - (V - N^* R_0)| = U - V + N^* R_0 \leq N^* R_0$. Besides, by the last paragraph, we have $U \geq V - N^* R_0 \geq \frac{1}{2} N^* L_0 - N^* R_0$. Thus, we have $\frac{|U-(V-N^*R_0)|}{U} = \frac{U-(V-N^*R_0)}{U} \leq \frac{N^* R_0}{\frac{1}{2}N^* L_0 - N^* R_0} = \frac{2R_0}{L_0 - 2R_0} = \frac{2\eta}{1-2\eta} = \sigma$, *i.e.*, $V - N^* R_0 \geq (1 - \sigma)U$. ■

By the definitions of N^* and R_0, we have that the net transmission rate corresponding to the selected solution of (12) is at least $V - N^* R_0$. Thus, we obtain the bound of the linear approximation compared with the primary MIP problem.

3.2.2 Bound of the FPTA Algorithm

We next analyze the bound of the FPTA algorithm. According to theorem in [19], we have the following lemma:

Lemma 3 If p is selected in each iteration to be the shortest (s_i, d_i) path among all commodities, then for a final flow value $W = \sum_{p \in \mathbf{P}} x(p)$ obtained from the FPTAS algorithm, we have $W \geq (1 - 2\epsilon)V$, where ϵ is the desired accuracy level.

Because the value of $x(p)$ is unchanged in our post-processes of Algorithm 1, W is also the final flow value of our proposed FPTA algorithm. Note that it is not the bound of the FPTA algorithm compared with the LP problem (10), because our objective function is the net transmission rate, while W is only the transmission rate of the solution of the FPTA algorithm. Besides, V is not the maximum net transmission rate as well. The bound of the FPTA algorithm is given in the following theorem:

Theorem 2 Let F be the net transmission rate corresponding to the solution of Algorithm 1. In the data center networks providing full bisection bandwidth, we have $F \geq (1 - 2\epsilon - \sigma)U$, where U is the optimal value of the primal MIP problem (8).

Proof: By the definitions of N^* and R_0, we have that the net transmission rate corresponding to the solution of the FPTA algorithm is at least $W - N^* R_0$, *i.e.*, $F \geq W - N^* R_0$. Thus we have $F \geq (1 - 2\epsilon)V - N^* R_0 = (1 - 2\epsilon)(V - N^* R_0) - 2\epsilon N^* R_0$. Meanwhile, by $U \geq \frac{1}{2} N^* L_0 - N^* R_0 \geq \frac{1}{2\eta} N^* R_0 - N^* R_0$, we have $N^* R_0 \leq \frac{2\eta}{1-2\eta} U = \sigma U$.

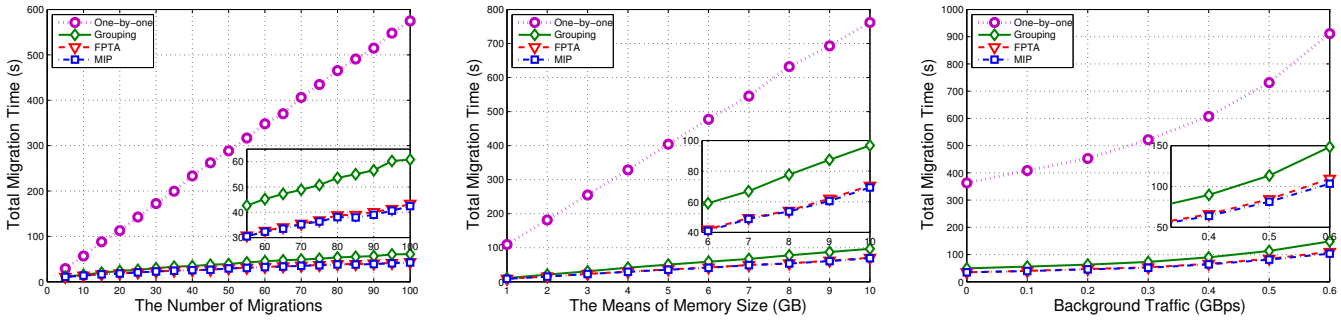


Fig. 5. Total migration time vs different parameters in one datacenter under the topology of PRV1.

By Theorem 1, we have $F \geq (1-2\epsilon)(1-\sigma)U - 2\epsilon\sigma U = (1-2\epsilon-\sigma)U$. Thus we obtain the bound of the FPTA algorithm compared with the primal MIP problem.■

3.3 Complexity Analysis

We now analyze the computational complexity of the FPTA algorithm. Since the computational time of the post-processes is only a linear function of the network size and the number of the VMs to be migrated, we mainly focus on the computational complexity of FPTAS algorithm to the maximum multicommodity flow problem. The dual to the problem (11) is

$$\begin{aligned} \textbf{max} \ & \textstyle\sum_{e\in\mathbf{P}} c(e)u(e) \\ \textbf{s.t.} \ & \begin{cases} \textstyle\sum_{e\in p} u(e) \geq 1, & \forall p \in \mathbf{P} \\ u(e) \geq 0, & \forall e \in \mathbf{E} \end{cases} \end{aligned} \tag{16}$$

According to Algorithm 1, at start, we have $l(e) = \delta$ for all $e \in \mathbf{E}$. The last time the length of an edge is updated, its length, $u(e)$ is less than one, and it is increased by at most a factor of $1+\epsilon$ in the update. Thus, the final length of any edge is less than $1+\epsilon$. Since every augmentation increases the length of the capacity of the minimum capacity edge by $1+\epsilon$, the number of augmentations is less than $m\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$, where m is the number of links in the network. Besides, there are $k\log_{1+\epsilon}\frac{1+\epsilon}{\delta}$ shortest path computations that do not lead to augmentations, where k is the number of the commodities, $i.e.$, the number of VMs to be migrated in our problem. Using the Dijkstra shortest path algorithm, the runtime of the algorithm is $O(\frac{1}{\epsilon^2}(m^2+km)\log n)$. This can be further reduced by grouping commodities by a common source node, of which the shortest path can be computed in one computation. More details can be found in [19]. Thus we have the following lemma.

Lemma 4 An ϵ-approximate maximum multicommodity flow can be computed in $O(\frac{1}{\epsilon^2}m(m+n\log m)\log n)$ time, where n is the number of nodes in the network and m is the number of links in the network.

Note that ϵ is not the desired accuracy level of our FPTA algorithm compared with the primal MIP problem. Then based on Lemma 4 and Theorem 2, the complexity of our proposed algorithm is given in the following theorem:

Theorem 3 An κ-approximate solution of the MIP problem (8) can be computed by the FPTA algorithm in $O(\frac{1}{(\kappa-\sigma/2)^2}m(m+n\log m)\log n)$ time.

Proof: According to Theorem 2, the accuracy level of the FPTA algorithm is $\epsilon+\sigma/2$. Thus, to obtain a κ-approximate solution of the primal MIP problem, we must ensure that $\epsilon+\sigma/2 \leq \kappa$, $i.e.$, $\epsilon \leq \kappa-\sigma/2$. Combining Lemma 4, we prove that an κ-approximate solution of the primal MIP problem (8) can be found in $O(\frac{1}{(\kappa-\sigma/2)^2}m(m+n\log m)\log n)$ time by the FPTA algorithm. ■

4 Performance Evaluation

4.1 Simulation System Set Up

With the increasing trend of owning multiple datacenter sites by a single company, migrating VMs across datacenters becomes a common scenario. Thus, to evaluate the performance of our proposed migration plan inside one datacenter and across datacenters, we select the following two topologies to implement our experiments: (1) The topology of a private enterprise data center located in Midwestern United States (PRV1 in [22]). There are 96 devices and 1088 servers in the data center network, which utilizes a canonical 2-Tier Cisco architecture. (2) B4, Google's inter-datacenter WAN with 12 data centers interconnected with 19 links [21] (showing in Fig. 2(a)). In B4, each node represents a data center. Besides, the network provides massive bandwidth. However, to evaluate the performance of our proposed algorithm under relatively hard conditions, we assume the capacity of each link is only 1GBps. On the other hand, the capacities of links in RPV1 are set ranging from 1GB to 10GB according to [22]. If not special specified, the page dirty rate of each VM is set to 100MBps. Besides, V_{thd} and T_r are set to 100MB and 20ms, respectively. The memory sizes of VMs are also set ranging from 1GB to 10GB unless stated otherwise.

We use a self-developed event-driven simulator to evaluate the performance of our proposed algorithm. Given the input, which includes the structure of the network, background traffic, migration requests of VM, page dirty rates, etc., the simulator will calculate the detailed process of migrations, and output the related metrics we need, including the total migration time, the total downtime and so on. Specifically, we compare the performance of our proposed FPTA algorithm with the optimal solution of the primary MIP problem (referred to as MIP algorithm) and two state-of-the-art algorithms. In the two state-of-the-art algorithms, one is the algorithm based on one-by-one migration scheme (referred to as one-by-one algorithm), which is proposed

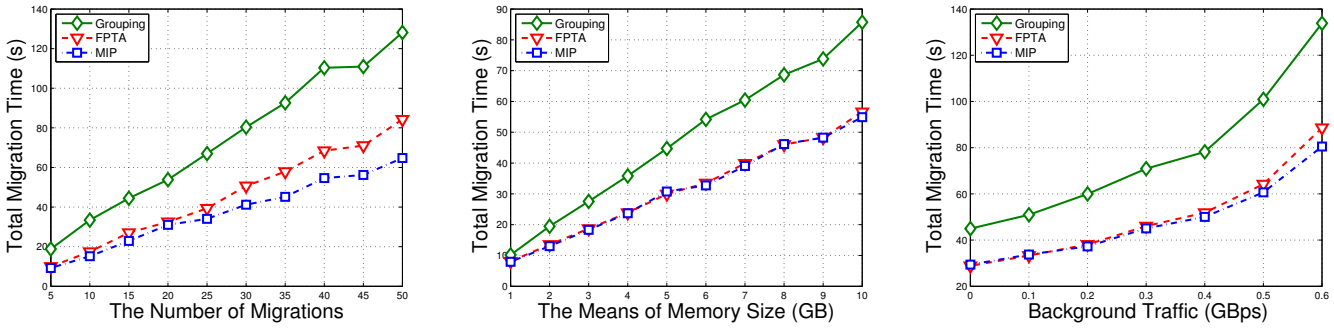


Fig. 6. Total migration time vs different parameters in inter-datacenter network under the topology of B4.

in [11], [35]. The other is the algorithm that migrates VMs by groups (referred to as grouping algorithm), just as the algorithm proposed in [9]. In this algorithm, VMs that can be migrated in parallel are divided into the same group, while VMs that share the same resources, such as the same link in their paths, are divided into different groups. Then VMs are migrated by groups according to their costs [9]. We further set the function of the cost as the weighted value of the total migration time and the number of VMs in each group.

4.2 Results and Analysis

4.2.1 Migration Time

In our first group of experiments, we assume all the VM migration requests, which are generated randomly, are known at the beginning. Given a group of VM migration requests, the total migration time, i.e., the time spent to finish all migrations, is evaluated as the metric of the performance. Specifically, we compare the performance of our proposed FPTA algorithm with that of other 3 algorithms introduced above, in PRV1 and B4 with different number of VMs to be migrated, different amount of background traffic, different average memory size of VMs, respectively. In addition, the volume of background traffic is generated randomly with specific mean value we set. The results are shown in Fig. 5 and Fig. 6.

As we can observe from the Fig. 5, the performance of the one-by-one algorithm is much worse than that of the other three algorithms: when there are 100 VMs to be migrated, the total migration time it takes is about 10 times more than that of the other three algorithms, illustrating its inefficiency in reducing the total migration time. Since the performance gap between one-by-one algorithm and the other algorithms is huge, we do not show its performance in Fig. 6.

As for the performance of the other three algorithms, their total migration time vs different parameters in data center networks and inter-datacenter WAN has a similar trend: the total migration time of FPTA algorithm is very close to that of the MIP algorithm, and much less than that of the grouping algorithm. Take Fig. 5(a) and Fig. 6(a) for example. In PRV1 (showing in Fig. 5(a)), total migration time of the FPTA algorithm and the MIP algorithm almost cannot be distinguished, while in B4 (showing in Fig. 6(a)) the gap is less than 15% relative to the MIP algorithm. Meanwhile, FPTA algorithm performs much better than the grouping algorithm: its migration time is reduced by 40% and 50% in comparison with the grouping algorithm in PRV1 and B4, respectively. Thus, the solution of our proposed FPTA algorithm approaches to the MIP algorithm and outperforms the state-of-the-art solutions.

4.2.2 Net Transmission Rate

To illustrate the effectiveness of maximizing the net transmission rate, we implement the second group of experiments in the scenario where there are 40 VMs to be migrated in B4. Net transmission rates, i.e., the sum of all migrating VMs minus their page dirty rates, of the FPTA algorithm and the grouping algorithm are evaluated, as functions of time. The result is shown in Fig. 7.

According to previous theoretic analysis, we know that the sum of memory sizes of VMs to be migrated is approximately equal to the integration of the net transmission rate with respect to time. In the experiments, the sum of memory sizes of the 40 VMs to be migrated are 203GB. Meanwhile, in Fig. 7, the shaded areas of the FPTA and grouping algorithm, which can represent the integrations of the net transmission rates with respect to time, are 203.0GB and 212.0GB, respectively. The relative errors are less than 5%. It proves the correctness of our theoretic analysis.

Besides, from the figure we observe that the net transmission rate with the FPTA algorithm remains a relatively high level in the process of migrations, about 2 times higher than that of the grouping algorithm on average. Thus, the integration of the net transmission rate can reaches $\sum_{k=1}^{40} m_k$ with less time. Specifically, in this group of experiments, the total migration time of FPTA algorithm is reduced by up to 50% compared with grouping algorithm. Thus, our FPTA algorithm significantly reduces the total migration time by maximizing the net transmission rate.

4.2.3 Computation Time

In this group of experiments, we simulate the computation time of the FPTA algorithm as a function of the number of VMs to be migrated in the inter-datacenter WAN. To be compared with, the computation time of using GLPK to solve the MIP problem of (8) and the LP problem of (10) is also simulated. The result is shown in Fig. 8. Since the time complexity of solving the primal MIP problem is too

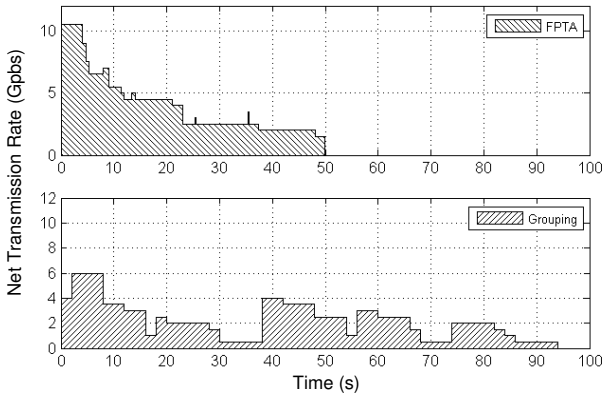


Fig. 7. The net transmission rates vs time of FPTA algorithm and grouping algorithm in inter-datacenter network under the topology of B4 with 40 VMs to be migrated.

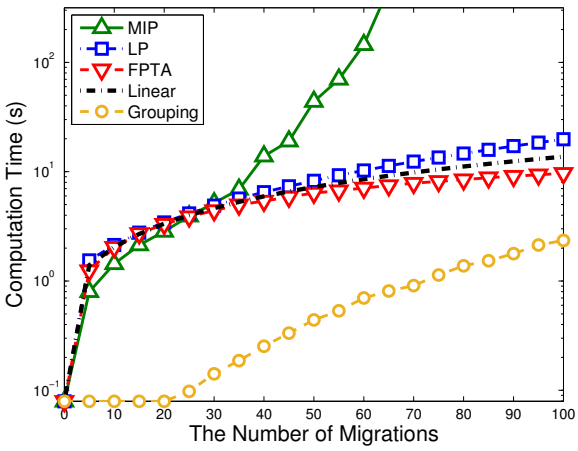


Fig. 8. Computation time of different algorithms

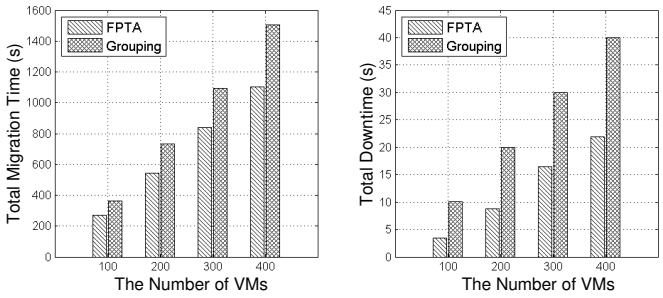
high, we use the logarithmic scale for the computation time in Fig. 8.

From the result we can observe that compared with solving the MIP problem directly, the FPTA algorithm significantly reduces the computation time. For example, when there are 60 VMs to be migrated, the computation time of the FPTA algorithm is only 6 seconds, while it takes 2.3 minutes to solve the primal MIP problem by GLPK. Besides, the computation time of the FPTA algorithm is uniformly less than that of solving the LP problem (10). When there are 100 VMs to be migrated, the FPTA algorithm reduces the computation time by 50% compared with solving the LP problem. What's more, we add a curve of the computational time of the linear computational complexity as a baseline in Fig. 8. We observe that the computation time of solving the MIP problem and the LP problem has a superlinear growth with respect to the number of VMs to be migrated, while the growth speed of the computation time of the FPTA algorithm is even less than the linear growth. The result agrees with the theoretic analysis in Section 3. However, as we can observe, the computation time of the grouping algorithm is the smallest due to its simple structure and greedy strategy, but we can also observe the superlinear growth of its computation time. Overall, the FPTA algorithm significantly reduces the computation time as well as guarantees a good performance.

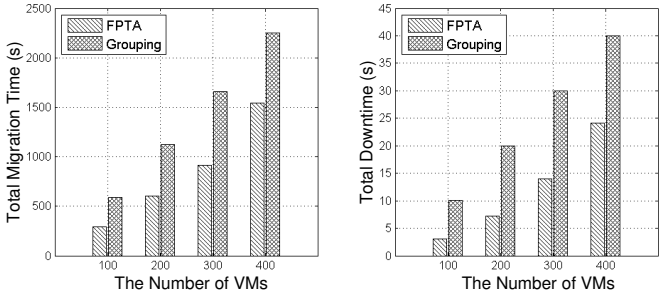
4.2.4 *Application Performance*

The scenarios of this group of experiments are to optimize the average delay of services in B4. Assume there are some VMs located randomly in the data centers in B4 at the beginning, and they are providing services to the same user, who is located closely to the node 8 (data center 8). Thus we need to migrate these VMs to data centers as close to the node 8 as possible. However, memory that each data center provides is not unlimited, which is set to range from 0.5TB to 1.5TB randomly in our experiments. Thus, only a part of VMs can be migrated to the node 8. To have the smallest average delay, we find the final migration sets. Then we use the FPTA and grouping algorithm to implement these migrations. The total migration time and the total downtime are simulated when there are 100, 200, 300, 400 VMs, respectively, in which the downtime is computed according to the mathematical model for live migration provided in section 2. In addition, we also evaluate the impact of the types of applications running on the VMs. Specifically, if the applications running on the VMs are data-heavy, such as database, video service, etc., their page dirty rate is relatively large. Similarly, if the applications running on the VMs are data-light, their page dirty rate is relatively small. We simulate the performance of our algorithm in these two conditions. The mean value of page dirty rate corresponding to the data-light application is set to be 0.1GBps, while for the data-heavy application, it is set to be 0.5GBps. In addition, we also assume that page dirty rate varies over time, and it ranges from 0.5 to 1.5 times relative to its mean. The results are shown in Fig. 9.

Fig. 9(a) and (b) show the performance of migration for VMs with data-light and data-heavy applications, respectively. From Fig. 9(a), we observe that the performance of the FPTA algorithm is uniformly better than that of the grouping algorithm. Specifically, compared with the grouping algorithm, FPTA algorithm reduces the total migration time by 25.6% on average. In addition, it also reduces the total downtime by 53.2% on average. As for the performance of the VMs with data-heavy applications, we can observe from Fig. 9(b) that the total migration time is obviously increased for both algorithms, and the total migration time of FPTA algorithm is still much smaller than that of grouping algorithm, i.e., reduced by up to 43.3%. On the other hand, the total downtime is not much influenced. The reason is that in order to maximize the net transmission rate of VMs with higher page dirty, our proposed FPTA algorithm reduces the number of concurrent migrated VMs, and preserves bandwidth for each single migration. Meanwhile, V_{thd}, the remaining dirty memory transferred at the last iteration, is fixed. Thus, the downtime, i.e., the time spent to transfer the final remaining dirty memory, is not much influenced. Overall, our proposed FPTA algorithm outperforms the grouping algorithm uniformly, which provides better services for the user.



(a) Data-light application (small page dirty rate)



(b) Data-heavy application (large page dirty rate)

Fig. 9. Total migration time and downtime for optimizing delay in inter-datacenter network under the topology of B4.

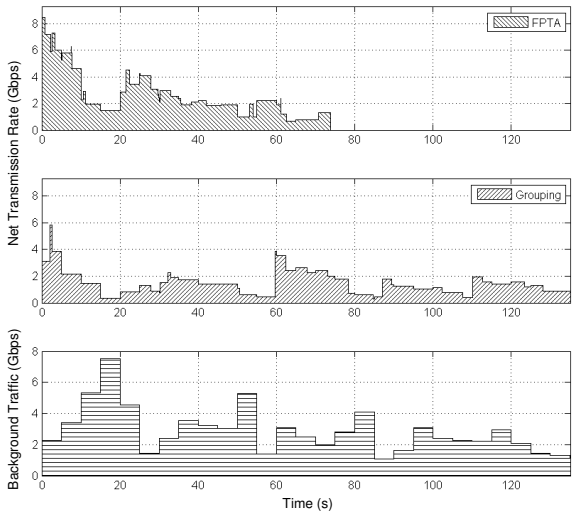


Fig. 10. The net transmission rates vs time of FPTA algorithm and grouping algorithm in inter-datacenter network under the topology of B4 with 40 VMs to be migrated.

4.2.5 Performance under Real Packet Trace

In the previous experiments, the background traffic is assumed to be time-invariant, and its volume is generated randomly. Thus, to evaluate the performance of our proposed algorithm with realistic traffic in data center, we conduct this group of experiments. The amount of the background traffic is obtained from the real packet-level traces from the data center of a university in Mid-United States, EDU1 provided in [22]. Then, we implement this group of experiments under the topology of B4 and PRV1 with the new obtained background traffic. For simplicity, in the experiments of the inter-datacenter WAN, each datacenter is regarded as a huge host, of which the internal structure is ignored. By elaborately setting the bandwidth of the local network interface of each host, the simplified problem can be equivalent to the original problem. Besides, we assume that expect for when there are some old migrations are finished, the SDN controller also dynamically adjusts the transmission rates of VMs every 5 seconds according to the mean amount of background traffic of the past 5 seconds.

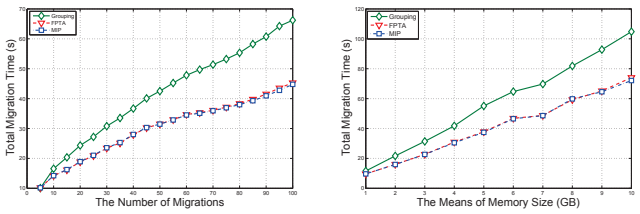
Similarly with the second group of experiments, the net transmission rates of the FPTA algorithm and the grouping algorithm are evaluated as functions of time under the topology of B4 with 40 VMs to be migrated. The result is shown in Fig. 10. Expect for the net transmission rates of the two algorithms, the curve of the volume of the background traffic is also plotted. As we can observe, compared with Fig. 7, the net transmission rates of different algorithms are also affected by the volume of the background traffic. For example, there is a peak of the background traffic at the $4th$ period of 5 seconds. In this period, the net transmission rates of both algorithms are at low points. Though affected by amount of the background traffic, the net transmission rate with the FPTA algorithm still remains a relatively high level in the process of migrations, about 2 times higher than that of the grouping algorithm on average, significantly reducing the total migration time by up to 45%. It indicates that the performance of our proposed algorithm with realistic traffic remains good.

Then, the total migration time of the FPTA and grouping algorithms is evaluated as a function of the number of VMs to be migrated and the average memory size of VMs in B4 and PRV1, respectively. The results are shown in Fig. 11. Still similarly with Fig. 5, the total migration time of FPTA algorithm is very close to that of the MIP algorithm, and much less than that of the grouping algorithm. Take Fig. 11(a) as an example. The migration time of the FPTA algorithm almost cannot be distinguished with that of the MIP algorithm, and is reduced by 30% on average in comparison with the grouping algorithm. Thus, the solution of our proposed FPTA algorithm approaches to the MIP algorithm and outperforms the state-of-the-art solution with realistic traffic.

5 Related Work

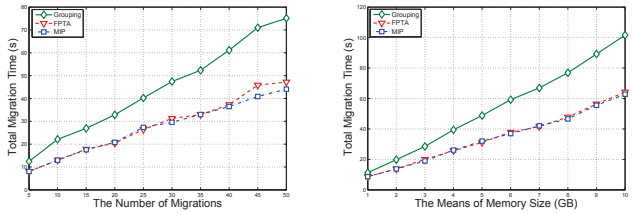
Works related to our paper can be divided by three topics: live migration, VM migration for improving quality and resilience of services, and migration planning.

Since Clark proposed live migration [2], there have been plenty of works that have been done in this field. Ramakrishnan $et\ al.$ [23] advocated a cooperative, context-aware approach to data center migration across WANs to deal with outages in a non-disruptive manner. Huang $et\ al.$ [24] presented a VM migration design by exploiting fast interconnects that support remote memory access technology to optimize migrations. Jin $et\ al.$ [25] proposed a VM migration approach using memory compression algorithms to optimize migrations. Michael $et\ al.$ [26] proposed post-copy based live migration for virtual machines, in which most



(a) Total migration time vs the number of VMs to be migrated in PRV1

(b) Total migration time vs the means of memory size in PRV1



(c) Total migration time vs the number of VMs to be migrated in B4

(d) Total migration time vs the means of memory size in B4

Fig. 11. Total migration time vs different parameters in one datacenter under the topology of B4 and PRV1.

pages are transferred after the VM's execution has been resumed at the target host. By using this method, the total migration time as well as downtime can be significantly reduced. Svärd *et al.* [27] compared existing approaches for live migration. From their results, it is seen that the pre-copy approaches are more robust, while post-copy approaches outperform in terms of resource usage and migration downtime. Wood *et al.* [28] presented a mechanism that provides seamless and secure cloud connectivity as well as supports live WAN migration of VMs. Nasim *et al.* [39] used multiplath-TCP (MPTCP) to efficiently aggregate the bandwidth of multiple disjoint paths in a model datacenter, and significantly reduce the downtime and migration time of VMs. In addition, Teka *et al.* [40] also presented an approach based on MPTCP to achieve seamless live VM migration, *i.e.*, not interrupting the application running on the VM, seamless connection migration and zero network VM downtime after the migration is completed. On the other hand, there have been a number of works about VM migration in SDNs. Mann *et al.* [29] presented CrossRoads – a network fabric that provides layer agnostic and seamless live and offline VM mobility across multiple data centers. Boughzala *et al.* [10] proposed a network infrastructure based on OpenFlow that solves the problem of inter-domain VM migration. Meanwhile, Keller *et al.* [31] proposed LIME, a general and efficient solution for joint migration of VMs and the network. Mann *et al.* [30] presented a QoS framework which uses a cost of migration model to allocate a minimal bandwidth for a migration flow such that it completes within the specified time limit while causing minimal interference to other flows in the network. These works indicate that SDN has big advantages in implementing VM migration. In contrast, we focus on developing a VM migration plan to reduce the total migration time in Software Defined Network (SDN) scenarios.

VM migration for improving quality or resilience of services has been studied in the literature [3]–[6], [32], [33]. Zhani *et al.* [5] focused on reducing network traffic to support bandwidth requirements and performance isolation among applications in data centers by migrating VMs to dynamically adjust the resource allocation. They proposed VDC Planner, a resource management framework for data centers, which aims at achieving high revenue while minimizing the total energy cost over-time. Wood *et al.* [6] presented Sandpiper, a system that automates the task of monitoring, detecting hotspots, and relocating VMs from overloaded to under-utilized nodes. Tsugawa *et al.* [3] studied the migration of multiple VMs for disaster recovery based on information collected after the Great East Japan Earthquake, and presented the condition to migrate VMs from damaged sites and keep IT services uninterrupted. The approaches of [32], [33] mainly focused on the replication of VMs rather than migration of VMs to improve the resilience of services, which consumes more storage resources to keep the replica and bandwidth resources to maintain synchronization between the replica and primary VMs. Fischer *et al.* [4] focused on the use of wide-area migration to increase the resilience of network service. They analyzed the application of VM migration or replication into the recovery of services, and presented two concepts for network redirection after the wide-area migration. Different with them, our work focus on the migration planning of VMs to reduce the total migration time.

Meanwhile, there have been some works about migration planning. However, most of them were designed under the model of one-by-one migration [11], [35] or their main focuses were not to optimize the total migration time [11], [34]. Ghorbani *et al.* [11] proposed a heuristic algorithm of determining the ordering of VM migrations and corresponding OpenFlow instructions. However, they concentrated on bandwidth guarantees, freedom of loops, and their algorithm is based on the model of one-by-one migration. Lo *et al.* [12] proposed three algorithms for migrating multiple virtual routers with the goal of minimizing the migration time and cost. Al-Haj *et al.* [34] also focused on finding a sequence of migration steps. They formulated their problem as a Constraints Satisfaction Problem (CSP) and used Satisfiability Modulo Theory (SMT) solvers to solve it. Their main goal was to satisfy security, dependency, and performance requirements. Another work [35] proposed an informed live migration strategy which considers the VMs' characteristics and the workload to determine the migration order. Hermenier *et al.* [36], [37] proposed a resource manager – Entropy, which performs dynamic consolidation based on constraint programming and takes migration overhead (cost) into account. However, optimizing the total migration time was still not their main goal. Table II summarizes the comparison of the approaches on migration planning on the basis of their objects to be migrated, target functions, and whether they enable multipath routing and migrating multiple objects simultaneously, where a dash will be displayed if the approach is not related with the corresponding problem.

TABLE 2
Comparison of approaches on migration planning.

Approach	*Object to be migrated*	*Enable multipath routing*	*Enable migrating multiple objects simultaneously*	*Target function*
[9]	VMs	No	Yes	Minimize the total migration time and service penalty due to migration
[11]	VMs	No	No	Meet constraints on bandwidth and loop-freedom
[12]	Virtual routers	No	Both	Minimize the migration time and cost
[34]	VMs	-	Yes	Satisfy security, dependency, and performance requirements
[35]	VMs	No	No	Improve the efficiency of reconfiguration a virtualized cluster
[36], [37]	VMs	-	Yes	Minimize the migration cost

6 Conclusion and Future Work

In this work, we focus on reducing the total migration time by determining the migration orders and transmission rates of VMs. Since solving this problem directly is difficult, we convert the problem to another problem, $i.e.$, maximizing the net transmission rate in the network. We formulate this problem as a mixed integer programming problem, which is NP-hard. Then we propose a fully polynomial time approximation (FPTA) algorithm to solve the problem. Results show that the proposed algorithm approaches to the optimal solution of the primary programming problem with less than 10% variation and much less computation time. Meanwhile, it reduces the total migration time and the service downtime by up to 40% and 20% compared with the state-of-the-art algorithms, respectively.

In future work, we plan to extend our algorithm to support the different priorities between migrations in the scheduling process. It can be realised by replacing the total net transmission rate in the objective function by the weighted summation of the net transmission rates of different VMs. In addition, it is also an interesting problem to extend our algorithm from migrating VMs to other network elements, such as containers, virtual routers, etc., which we leave as a future work. Besides, we assume that the SDN controller dynamically adjusts the transmission rates of VMs every 5 seconds according to the mean amount of background traffic of the past 5 seconds in this work. We plan to study more about dynamic performance of our proposed algorithm to determine a more appropriate dynamic model in future work.

Acknowledgment

This work is supported by National Basic Research Program of China (973 Program) (No. 2013CB329105), National Nature Science Foundation of China (No. 61301080, No. 61171065 and No. 61273214), National High Technology Research and Development Program (No. 2013AA013501 and No. 2013AA013505), Chinese National Major Scientific and Technological Specialized Project (No. 2013ZX03002001), Chinas Next Generation Internet (No. CNGI-12-03-007). Part of this work was presented as a paper at the $34th$ IEEE International Conference on Computer Communications (INFOCOM 2015) [41].

References

[1] P. Barham, B. Dragovic, K. Fraser, "Xen and the art of virtualization", *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164-177, 2003.
[2] C. Clark, K. Fraser, and S. Hand, "Live migration of virtual machines", in *Proc. 2nd NSDI*, 2005, pp. 273–286.
[3] M. Tsugawa, R. Figueiredo, J. Fortes, "On the use of virtualization technologies to support uninterrupted IT services: A case study with lessons learned from the Great East Japan Earthquake", in *Proc. of IEEE ICC*, 2012, pp. 6324–6328
[4] A. Fischer, A. Fessi, G. Carle, "Wide-area virtual machine migration as resilience mechanism", in *Proc. of IEEE SRDSW*, 2011, pp. 72–77.
[5] M. F. Zhani, Q. Zhang, G. Simona, "VDC Planner: Dynamic migration-aware virtual data center embedding for clouds", in *Proc. of IFIP/IEEE IM*, 2013, pp. 18-25.
[6] T. Wood, P. J. Shenoy, A. Venkataramani, "Black-box and Gray-box Strategies for Virtual Machine Migration", in *NSDI*, 2007, pp. 17.
[7] K. Ye, X. Jiang, D. Huang, "Live migration of multiple virtual machines with resource reservation in cloud computing environments" in *Proc. of IEEE CLOUD*, 2011, pp. 267-274.
[8] C. Mastroianni, M. Meo, G. Papuzzo, "Self-economy in cloud data centers: Statistical assignment and migration of virtual machines", *Euro-Par 2011 Parallel Processing*. Springer Berlin Heidelberg, 2011, pp. 407-418.
[9] M. F. Bari, M. F. Zhani, Q. Zhang Q, "CQNCR: Optimal VM Migration Planning in Cloud Data Centers", in *IFIP Networking Conference*, 2014, pp. 1-9.
[10] B. Boughzala, R. Ben Ali, M. Lemay, "OpenFlow supporting inter-domain virtual machine migration", in *Proc. of IEEE/IFIP WOCN*, 2011, pp. 1-7.
[11] S. Ghorbani, M. Caesar, "Walk the line: consistent network updates with bandwidth guarantees", in *Proc. of HotSDN*, 2012, pp. 67-72.
[12] S. Lo, M. Ammar, E. Zegura, "Design and analysis of schedules for virtual network migration", in *IFIP Networking Conference*, 2013, pp. 1-9.
[13] S. Agarwal, M. Kodialam, T. V. Lakshman, "Traffic engineering in software defined networks", in *Proc. of IEEE INFOCOM*, 2013, pp. 2211-2219.
[14] A. Sridharan, R. Guerin, C. Diot, "Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks", *IEEE/ACM Transactions on Networking (TON)*, vol. 13, no. 2, pp. 234-247, 2005.
[15] A. Tootoonchian, M. Ghobadi, Y. Ganjali, "OpenTM: traffic matrix estimator for OpenFlow networks", in *International Conference on Passive and Active Network Measurement*, Springer Berlin Heidelberg, pp. 201-210, 2010.
[16] J. Lofberg, "YALMIP: A toolbox for modeling and optimization in MATLAB", in *IEEE CACSD*, 2004, pp. 284-289.
[17] A. Makhorin, "GLPK (GNU linear programming kit)", 2008.
[18] N. Giorgetti, N. Klitgord, "Glpkmex: A matlab mex interface for the glpk library", 2009.
[19] L. K. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities", *SIAM Journal on Discrete Mathematics*, vol. 13, no. 4, pp. 505-520, 2000.
[20] D. G. Luenberger, Y. Ye, "Linear and nonlinear programming", *Springer*, 2008.

[21] S. Jain, A. Kumar, S. Mandal, "B4: Experience with a globally-deployed software defined WAN", in *Proc. of ACM SIGCOMM*, 2013, pp. 3-14.

[22] T. Benson, A. Akella, D. A. Maltz, "Network traffic characteristics of data centers in the wild", in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 267-280.

[23] K. K. Ramakrishnan, P. Shenoy, J. Van der Merwe, "Live data center migration across WANs: a robust cooperative context aware approach",in *Proceedings of the 2007 SIGCOMM workshop on Internet network management*", 2007, pp. 262-267.

[24] W. Huang, Q. Gao, J. Liu, "High performance virtual machine migration with RDMA over modern interconnects",in *2007 IEEE International Conference on Cluster Computing*, 2007, pp.11-20.

[25] H. Jin, L. Deng, S. Wu, "Live virtual machine migration with adaptive, memory compression",in *2009 IEEE International Conference on Cluster Computing and Workshops*, 2009, pp. 1-10.

[26] M. R. Hines, U. Deshpande, K. Gopalan, "Post-copy live migration of virtual machines", *ACM SIGOPS operating systems review*, vol. 43, no. 3, pp. 14-26, 2009.

[27] P. Svärd, B. Hudzia, S. Walsh, J. Tordsson, E. Elmroth, "Principles and performance characteristics of algorithms for live VM migration", *ACM SIGOPS Operating Systems Review*, vol. 49, no 1, pp. 142–155, 2015.

[28] T. Wood, K. K. Ramakrishnan, P. Shenoy, "CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines",in *ACM SIGPLAN Notices*, 2011, pp. 121-132.

[29] V. Mann, A. Vishnoi, K. Kannan, "CrossRoads: Seamless VM mobility across data centers through software defined networking", in *Proc. of IEEE NOMS*, 2012, pp. 88-96.

[30] V. Mann, A. Vishnoi, A. Iyer, and P. Bhattacharya, "VMPatrol: Dynamic and Automated QoS for Virtual Machine Migrations", in *Proc. of 8th CNSM*, 2012, pp. 174-178.

[31] E. Keller, S. Ghorbani, M. Caesar, "Live migration of an entire network (and its hosts)", in *Proc. of HotNets*, 2012, pp. 109-114.

[32] S. K. Bose, S. Brock, R. Skeoch, "Optimizing live migration of virtual machines across wide area networks using integrated replication and scheduling," in *Proc. of IEEE SysCon*, 2011, pp. 97–102.

[33] B. Cully, G. Lefebvre, D. Meyer, "Remus: High availability via asynchronous virtual machine replication," in *Proc. of NSDI*, 2008, pp. 161–174.

[34] S. Al-Haj, E. Al-Shaer, "A formal approach for virtual machine migration planning", in *CNSM*, 2013, pp. 51-58.

[35] X. Li, Q. He, J. Chen, K. Ye, and T. Yin, "Informed live migration strategies of virtual machines for cluster load balancing", in *Proc. of NPC 2011*, (Berlin, Heidelberg), 2011, pp. 111-122, Springer-Verlag.

[36] F. Hermenier, X. Lorca, J. M. Menaud, "Entropy: a consolidation manager for clusters", in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, 2009, pp. 41-50.

[37] F. Hermenier, A. Lèbre, J. M. Menaud, "Cluster-wide context switch of virtualized jobs", in *proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, 2010, pp. 658-666.

[38] M. Al-Fares, A. Loukissas, A. Vahdat, "A scalable, commodity data center network architecture", in *ACM SIGCOMM Computer Communication Review*, 2008, pp. 63-74.

[39] R. Nasim, A. J. Kassler, "Network-centric Performance Improvement for Live VM Migration," in *Proc. of IEEE CLOUD*, 2015, pp. 106–113.

[40] F. Teka, C. H. Lung, S. Ajila, "Seamless Live Virtual Machine Migration with Cloudlets and Multipath TCP," in *Proc. of COMPSAC*, 2015, vol. 2, pp. 607–616.

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