This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2871118, IEEE Transactions on Cloud Computing

Virtual Network Recognition and Optimization in SDN-enabled Cloud Environment

He Li, Kaoru Ota, Mianxiong Dong,
Department of Information and Electronic Engineering, Muroran Institute of Technology,
Muroran, Hokkaido, Japan.
E-mail: {heli, ota, mxdong}@mmm.muroran-it.ac.jp

Abstract—Cloud computing is a scalable and efficient technology for providing different services. For better reconfigurability and other purposes, users build virtual networks in cloud environments. Since some applications bring heavy pressure to cloud datacenter networks, it is necessary to recognize and optimize virtual networks with different applications. In some cloud environments, cloud providers are not allowed to monitor user private information in cloud instances. Therefore, in this paper, we present a virtual network recognition and optimization method to improve quality-of-service (QoS) of cloud services. We first introduce a community detection method to recognize virtual networks from the cloud datacenter network. Then, we design a scheduling strategy by combining SDN-based network management and instance placement to improve the service-level agreements (SLA) fulfillment. Our experimental result shows that we can achieve a recognition accuracy as high as 80% to find out the virtual networks, and the scheduling strategy increases the number of SLA fulfilled virtual networks.

Index Terms—Software Defined Networking, Network Virtualization, Cloud Computing, Datacenter Networks

1 Introduction

Recent years, more and more users deploy different applications in the cloud environment with better flexibility and scalability. For example, many users purchase big data cloud services such as Amazon EMR and Google Clod Dataproc to build their big data applications. However, if users want to use self-configurable platforms for specific purposes, they have to choose infrastructure-as-a-service (IaaS) instead of default big data services. For example, for graphics processing units (GPGPU) accelerated services, users usually need to build the platform themselves and purchase GPU resources from cloud providers [1], [2].

Many cloud computing scheduling methodologies focus on virtual network optimizations [3]. One method is instance placement optimization. The cloud scheduler adjusts the placement of each instance for different purposes [4]. Another method is adjusting network parameters of each virtual network, such as network priority, forwarding path and available bandwidth [5]. Some previous works present combinations of different methods for optimizing the cloud service performance [6]. In existing virtual network optimization techniques, a prerequisite is that the cloud scheduler knows the virtual topologies.

Cloud providers usually have efficient scheduling tools to optimize the quality of service (QoS) of commercial cloud services [7]. When cloud users build their virtual networks in the cloud platform, the cloud provider can record all configurations then optimize the performance with their scheduling tools. However, it is still hard to provide an efficient scheduling method without enough configuration information of virtual networks for general-purpose applications. Usually, the cloud provider is not allowed to access the data or other information in IaaS instances [8] to understand the virtual network configurations. Therefore, the organization of virtual networks in IaaS cloud is agnostic to conventional scheduling methods.

Therefore, in this paper, we first study virtual networks in the cloud environment. Usually, there are some methods to investigate instance relationships from the resource consumption. Software defined networking (SDN) technology is an emerging approach to allow the cloud provider monitor the end-to-end network traffic between IaaS instances [9]. Therefore, we choose network traffic records for the virtual network recognition. Since a virtual network in the datacenter network is like a instance community in a complex network, the virtual network recognition problem can be equivalent to a community detection problem. Thus, we introduce community detection model to identify the virtual networks from the end-to-end network traffic records.

After recognizing virtual networks, we design a scheduling strategy to optimize the QoS of cloud computing. In cloud computing, for scheduling limited resources for instances, instance placement is an acceptable solution that places instances near to required resource. However, instance placement will bring additional overheads in migration and resource reallocation. Because of the flexible management of the SDN-enabled network, it is possible to adjust network resources with negligible overhead. Therefore, we combine SDN-based management and instance placement to improve the scheduling efficiency. Furthermore, we choose the user

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2871118, IEEE Transactions on Cloud Computing

service-level agreements (SLA) fulfillment [10], a typical QoS measurement of cloud computing, as the scheduling object instead of other performance metrics. Finally, we evaluate our work by extensive simulations and the results show our method performs better than other optimizations.

The main contributions of this paper are summarized as follows.

- We first study the virtual network recognition in SDN-enabled cloud computing environment. We propose a community detection method to recognize virtual networks from end-to-end network traffic information recorded in the SDN controller.
- We propose a QoS-aware scheduling algorithm to improve the SLA fulfillment of cloud services based on the recognized virtual networks. We combine the network management and instance placement in the scheduling algorithm.
- We evaluate the recognition and optimization method by extensive simulations with settings and traffic trace data from real-world cloud computing networks. We also compare our work with the other optimizations.

The rest of this paper is summarized as follows. Section 2 reviews the related work. Our network scenario and motivations are introduced in Section 3. Section 4 presents the recognition model. A QoS-aware scheduling is proposed in Section 5. Section 6 gives the simulation results. Finally, Section 7 concludes this paper and give the future work.

2 Related Work

In this section, we first discuss the network scheduling strategies in the cloud environment. Then, we introduce some related SDN technologies for cloud computing.

2.1 Network Scheduling in Cloud Environment

There are some previous works focusing on network scheduling in the cloud environment. The network resource is a critical resource since all physical servers are interconnected by datacenter networks, and communication overhead constrains overall performance. There are mainly two aspects on improving network performance in the cloud environment. One is to design an appropriate network topology, which is determined before the deployment of cloud systems. Usually, most cloud datacenters choose tree-like topologies such as fat trees or hypercubes. These topologies focus on increasing the number of network ports, which linearly increases the network bandwidth. However, changing the network topology in an existing cloud environment is difficult. Thus, another aspect that scheduling network resource is more feasible in cloud computing.

Job scheduling is an efficient method for optimizing network performance by adjusting the Job arrangement between different cloud servers. Wang et al. [11] proposed a task management system to improve the QoS of cloud services in a Service-as-a-Service (SaaS) environment. From their work, the network management is an important issue for improving QoS of cloud jobs. The authors also introduced the task management system into a hybrid environment consisting of local and remote clouds [12]. Moreover, Erol et al. [13] implemented a task management system into EU FP7 PANACEA project, and the experimental results show that the task management system improved the job execution efficiency in a real-world testbed. However, for an IaaS cloud, the cloud provider is not allowed to manage user tasks directly.

A traditional solution is network overprovisioning which brings an unacceptable cost to large scale datacenter networks. Moreover, since the detailed traffic models are hard to be known, it is not appropriate for the cloud environment. QoS-policies-based service differentiation is an emerging method that segregates traffic for isolating performance to permit traffic engineering [14]. An important solution is network virtualization that introduces virtual networks for deployment of user customized networks [15]. Thus, in this paper, we focus on user virtual networks in the cloud environment.

Instance placement is a major solution for scheduling resources in the cloud environment. Meng et al. [16] proposed a heuristic method to optimize the aggregate traffic rates in the cloud datacenter network by placing cloud instances together with large mutual traffic. Jayasinghe et al. [17], Shrivastra et al. [18] and Wen et al. [19] also introduced similar approaches to minimize cloud datacenter traffic. From the production traces based experiments, these methods show a significant improvement for the network performance. Biran et al. [20] proposed a heuristic solution that jointly optimizes the routing between physical servers and instance placement, which brings better performance on network scheduling. Alicherry et al. [21] focus on the geo-distributed cloud environment and place instances across different datacenter networks. Hu et al. [22] presented vBundle system focusing on instance placement in user groups to optimize traffic between physical servers. Since instance placement brings additional overhead to the cloud system, we combined network management and instance placement together in our scheduling strategy.

2.2 SDN-enabled Cloud Computing

SDN is a key network technology for service provisioning of network services in the cloud environment [23]. SDN can provide flexible and efficient network resource management for supporting on-demand cloud services. For example, Frederic et al. [24] proposed a SDN-based cognitive packet network algorithm to optimize the routing management, which is a potential solution for cloud datacenter network optimization. Furthermore, the authors introduced the cognitive routing engine into the cloud environment to improve the QoS of the tenant network performance [25].

Cloud data center networks need several fundamental requirements, including scalable deployment, dynamic

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2871118, IEEE Transactions on Cloud Computing

resource provisioning, QoS differentiation, and flexible network control [26]. Thus, many works introduce SDN to meet these requirements and there are two important issues for SDN-enabled cloud computing. The first issue is namely virtual switching that builds communication paths between instances in the same physical server. Original virtual switching solutions can not provide enough visibility and management on the communication in the same physical server. OpenvSwitch, a software implementation of the SDN-enabled switch, is introduced as edge switching to provide visible and manageable instance communication in the same physical server. Cloud providers can collect network status and traffic information from these software SDN devices [27], [28]. Moshref et al. [29] proposed a virtual cloud rule information base (vCRIB) to improve OpenvSwitch performance in the cloud environment that optimizes rule placement among physical and virtual switches.

Instance migration is another application of SDN-enable cloud computing [30]. Since the IP protocol limits the broadcast domain in datacenter networks, it is hard to migrate instance across different broadcast domains. Address resolution protocol (ARP) also limits the instance addressing in the broadcast domain. SDN-enabled cloud computing provides new opportunity to solve this problem by introducing mobile IP and locator/identifier separation protocol (LISP) into instance migration [31], [32], [33]. OpenDaylight, the largest open source SDN controller, has added a LISP-based mapping service for support instance migration in the cloud environment [34]. After inserting forwarding rules in SDN devices, it is possible to support intra- and inter- datacenter migration [35]. Mann et al. [36] proposed an instance migration method named CrossRoads that migrates instances across cloud datacenters using pseudo address mapping in the SDN controller. For maintaining instance connections after instance migration, CrossRoads changes the forwarding rules and then sets the correct destination. Due to the opportunities brought by SDN technologies, we propose the SDN-based virtual network recognition and bandwidth adjustment in this paper.

3 Background and Motivation

In this section, we first discuss the virtual networks in the cloud environment and then introduce the SDN-enabled cloud datacenter network for the recognition and optimization methodology.

3.1 Virtual Network in Cloud Computing

We focus on the network virtualization in an IaaS environment. As shown in Fig. 1, there is a cloud datacenter network with eight physical servers denoted by p_1 to p_8. Instances are executed on these servers and connected by the tree network. In user configurations, there are three virtual networks denoted by v_1, v_2, and v_3. We assume the cloud administrator places these instances accordingly to a default scheduling strategy.

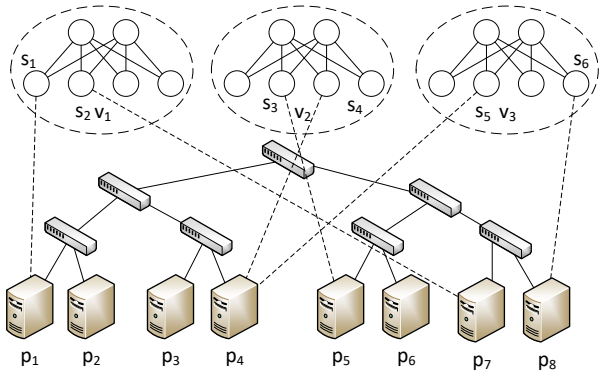


Fig. 1. Virtual networks in a cloud datacenter network

We choose several instances to describe the scheduling problem, which are denoted by s_1 to s_6, and assign s_1 and s_2 to virtual network v_1, s_3 and s_4 to v_2, and s_5 and s_6 to v_3. The cloud administrator puts instance s_1 on physical service p_1, s_2 on p_7, s_3 on p_5, s_4 and s_5 on p_4, and s_6 on p_8. Thus, when users execute applications in these virtual networks, the data flows between nearby instances in virtual networks will be forwarded across multiple switches and routers in the data center networks.

Since most data center networks choose tree-like topologies, the traffic across different aggregation switches is heavy. In traditional clusters or private clouds, administrators are able to adjust instance placement to resolve this problem with virtual network configurations. A possible adjustment is to place all instances from the same virtual network into a subnet under the aggregation switch. In our example, placing instance s_2 to server p_1, s_4 to p_5, and s_5 to p_8 could solve the problem.

However, the administrator is not allowed to access the private information in instances. Furthermore, an appropriate scheduling strategy for different applications is another challenge. Therefore, in the rest of this paper, we will discuss a new methodology focusing on recognizing and optimizing virtual networks in the cloud environment.

3.2 SDN-enabled Cloud Datacenter Network

Since we chose network traffic information for the virtual network recognition, a convenient way to monitor network information is using SDN. Thus, we introduce the SDN-enabled cloud datacenter network as the main scenario of our methodology.

As shown in Figure 2, there is a small cloud data center network with a SDN controller. Cloud servers connected by a tree topology network and all network devices are connected to the SDN controller. In the management server, the cloud controller can migrate instances for load balance and scheduling strategies.

In the SDN-enabled cloud datacenter network, we assume that all network devices including virtual and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2871118, IEEE Transactions on Cloud Computing

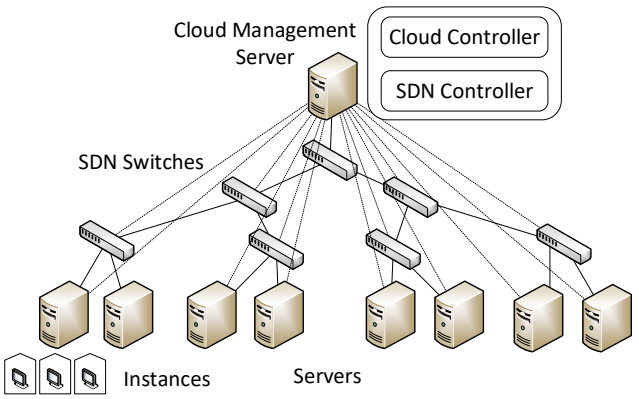


Fig. 2. SDN-enabled Cloud Datacenter Network Structure

TABLE 1
Example of the end-to-end traffic information

10/11/2016, 3:43 AM (UTC)						
-	s_1	s_2	s_3	s_4	s_5	s_6
s_1	-	3738	0	0	0	0
s_2	4670	-	0	0	0	0
s_3	0	0	-	1570	0	0
s_4	0	0	1066	-	0	0
s_5	0	0	0	0	-	5312
s_6	0	0	0	0	6230	-

physical network devices support SDN protocols. Thus, the SDN controller can monitor end-to-end traffic information of all network links. The traffic information is stored in the cloud management server for further scheduling strategies. For protecting user privacy, we assume that the controller has no authority to get private data in network flows. Only the number of packets transferred in network flows is transferred to the SDN controller.

As shown in Table 1, we can get the end-to-end traffic information from the SDN-enabled cloud datacenter network. The value in each cell is the number of packets transfered in end-to-end flows. There are only a few packets across different virtual networks because of the network isolation. Thus, it is possible to recognize virtual networks by analyzing the end-to-end traffic information.

4 Virtual Network Recognition

In this section, we discuss the method for recognizing virtual networks in the cloud environment.

There are some solutions for identifying virtual networks with the network traffic. Since the virtual network can be considered as a group of instances, the virtual network recognition is similar to the community detection problems in complex networks [37]. Thus, we formulate the virtual network recognition as a community identification problem and design an efficient algorithm to solve it.

In the recognition problem, we consider the network traffic in each link as the weight of an edge. We formulate the problem as a community identification in a weighted network. Let $\mathbf{G}$ denote the entire network of all instances, $\mathbf{V}$ denote the instance set, and $\mathbf{E}$ denote the edge set between instances, in which $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. For two instance $a, b \in \mathbf{V}$, we use a two-tuples group $(a, b) \in \mathbf{E}$ to denote the edge, and w_{ab} and w_{ba} to denote the network traffic between instance a and b, respectively. The set of virtual networks is denoted by O and a virtual network is denoted by $o_i, i \in [1, |O|]$. For an instance $a \in \mathbf{V}$, we use k_a to denote the degree and N_a to denote neighbor instance set of instance a, where $k_a = \sum_{b \in N_a}(w_{ab} + w_{ba})$. We use a belonging degree $B(a, o_i)$ to denote the relationship between instance a and virtual network o_i, given by

$$B(a, o_i) = \frac{\sum_{b \in o_i}(w_{ab} + w_{ba})}{k_a} \tag{1}$$

where all neighbor instances of node a are included in virtual network o_i, $B(a, o_i) = 1$.

Since each virtual network has heavier network traffic between member instances, it is necessary to identify the community which has more internal traffic. Thus, we adopt the normalized cut metric, which is a notion in community goodness [38]. Let $\Phi(o_i)$ to denote the normalized cut metric of virtual network $o_i \in O$, given by

$$\Phi(o_i) = \frac{\mathrm{cut}(o_i, \mathbf{G} \backslash o_i)}{w_{o_i}} \tag{2}$$

where $\mathrm{cut}(o_i, \mathbf{G} \backslash o_i)$ is the traffic of the cut edges of o_i and w_{o_i} is the traffic of all edges in community o_i.

Since the identification accuracy is better with more internal traffic and lower normalized cut metric, we optimize the $\Phi(o_i)$ in our virtual network recognition. Unfortunately, the normalized cut metric optimization is NP-hard in community detection [38]. Thus, we design a heuristic algorithm to identify virtual networks.

As shown in Algorithm 1, the algorithm first sets O as empty. Then, the algorithm begins a loop to find all virtual networks in network $\mathbf{G}$. In the loop, an initial edge (a, b) with the heaviest traffic is found and instance a and b on this edge are put into virtual network o_i. Based on the initial virtual network, the algorithm searches the set of adjacent instances denoted by N_{o_i}. Then, the algorithm traverses all adjacent instances to extend the virtual network. If adding an adjacent instance d into the virtual network can bring lower normalized cut metric, the algorithm adds the new instance into virtual network o_i and updates the adjacent instances of o_i. After identifying a new virtual network o_i, the algorithm deletes all edges in virtual network o_i and adds the virtual network into the result.

5 Virtual Network Optimization

In this section, we describe our scheduling strategy for optimizing QoS of the cloud system. We first statement

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2871118, IEEE Transactions on Cloud Computing

Algorithm 1 Virtual network recognition algorithm

$O \leftarrow \varnothing$;
while $\mathbf{E} \neq \varnothing$ **do**
  $(a,b) \leftarrow \arg\max_{(a,b)\in\mathbf{E}}(w_{ab}+w_{ba})$;
  $o_i \leftarrow \{a,b\}$;
  $N_{o_i} \leftarrow \varnothing$;
  for instance $h \in o_i$ **do**
    $N_{o_i} \leftarrow N_{o_i} \cup N_h$
  end for
  while $N_{o_i} \neq \varnothing$ **do**
    $c_i' \leftarrow o_i \cup \arg\max_{f\in N_{o_i}} B(f, o_i)$;
    if $\Phi(c_i') \leq \Phi(o_i)$ **then**
      $o_i \leftarrow c_i'$;
    end if
    $N_{o_i} \leftarrow \varnothing$
    for instance $h \in o_i$ **do**
      $N_{o_i} \leftarrow N_{o_i} \cup N_h$
    end for
  end while
  $E \leftarrow E \backslash E_{o_i}$;
  $O \leftarrow O \cup o_i$;
end while

TABLE 2
Notations in the virtual network optimization problem

Notation	Description
M	Set of physical servers
m_i	Server in set M
V	Set of virtual networks in the recognition result
v_j	Virtual network in set V
C	Available bandwidth matrix of all links between servers
$c_{ii'}$	Available bandwidth between server m_i and m_i'
S_j	Set of instances in virtual network v_j
s_{jk}	Instance in set S_j
L_j	Placement matrix of all instances in set S_j
l_{ijk}	Relationship between instance s_{jk} and server m_i
r^d_{jk}	Required resource of instance s_{jk}
$c^d_{jkk'}$	Required bandwidth between instance s_{jk} and $s_{jk'}$
r^a_{jk}	Assigned resource for instance s_{jk}
$c^a_{jkk'}$	Assigned bandwidth for the link between instance s_{jk} and $s_{jk'}$
R^c_i	Resource capacity of server m_i
Q_j	SLA fulfillment ratio of virtual network v_j
$F(\cdot)$	Monotone increasing function of all r^a_{jk} and $c^a_{jkk'}$

the virtual network optimization problem as a non-cooperative game and then give a Pareto efficient solution.

5.1 Problem Statement

We use M to denote the set of physical servers, and $m_i, i \in [1,|M|]$ to denote a server in set M. Then, we use V to denote the set of all virtual networks in the recognition result, and $v_j, j \in [1,|V|]$ to denote one virtual network in set V.

Since there are different topologies and settings of datacenter networks, we use an elastic $|M|\times|M|$ matrix C to denote the available bandwidth between each server. In matrix C, let $c_{ii'}, i \in [1,|M|], i' \in [1,|M|]$ denote the bandwidth between server m_i and m_i'. We also define a set S_j to denote all instance in virtual network v_j and $s_{jk}, k \in [1,|S_j|]$ denote an instance in set S_j.

In the instance placement, since most cloud datacenters choose shared storage solutions for storing instance data, the cost of instance migration is mainly incurred in transferring memory data. We use a function $l(s_{jk}, m_i, m_i^0)$ to denote the cost for placing instance s_{jk} on server m_i, where m_i^0 is the original position of instance s_{jk}. We also use a value L_{ijk} to denote the placement strategy, given by

$$l_{ijk} = \begin{cases} 1, & \text{if instance } s_{jk} \text{ is placed on } m_i, \\ 0, & \text{ohterwise.} \end{cases} \tag{3}$$

Then, we study QoS measurement in the cloud environment. In cloud computing, the cloud provider will use SLA fulfillment to measure the QoS. Usually, SLA is designed for performance guarantee or resource consumption of applications. In our scenario, we choose SLA as an agreement on performance guarantee of the cloud system. Thus, the SLA fulfillment is an appropriate way to measure the QoS of cloud applications. Actually, the SLA fulfillment is a satisfaction ratio from each user according to the task completion time. If the task is accomplished before the time in the SLA, the fulfillment is 100% otherwise less than 100%.

Therefore, the QoS can be measured by the task completion time. We assume the default completion time is confirmed by the original configuration. We also assume the required bandwidth is predictable after virtual network recognition. We use r^d_{jk} to denote the required resource of instance s_{jk}, and $c^d_{jkk'}, k' \in [1,|S_j|]$ to denote the predict required bandwidth between instance s_{jk} and $s_{jk'}$. The assigned resource of instance s_{jk} is denoted by r^a_{jk}, and assigned bandwidth between instance s_{jk} and $s_{jk'}$ is denoted by $c^a_{jkk'}$. The resource capacity of server m_i is denoted by R^c_i.

Thus, the placement strategy should satisfy the constraints as

$$\sum_{j=1}^{|V|}\sum_{k=1}^{|S_j|} r^a_{jk} \cdot l_{ijk} \leq R^c_i, \text{ and} \tag{4}$$

$$\sum_{j=1}^{|V|}\sum_{k=1}^{|S_j|}\sum_{k'=1}^{|S_j|} c^a_{jkk'} l_{ijk} l_{i'jk'} \leq c_{ii'} \tag{5}$$

where $i \neq i'$ and $k \neq k'$.

Then, we calculate the SLA fulfillment ratio of virtual network v_j. We use Q_j to denote the SLA fulfillment, given by

$$Q_j = \begin{cases} 100\%, & \text{if } \forall k \in [1,|S_j|], r^a_{jk} \geq r^d_{jk} \\ & \text{and } \forall k' \in [1,|S_j|], c^a_{jkk'} < c^r_{jkk'} \\ F(r^a_{j*}, c^a_{j**'}), & \text{otherwise,} \end{cases} \tag{6}$$

where $0 \leq F(\cdot) < 1$ is an elastic function for measuring SLA fulfillment ratio from r^a_{jk} and $c^a_{jkk'}$, $\forall k \in [1,|S_j|]$ and $k \in [1,|S_j|]$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2871118, IEEE Transactions on Cloud Computing

Since we focus on the network resource scheduling, we assume that the assigned resource of each instance is equal to required resource. We use $f_{jkk'}$ to denote the total traffic between instance s_{jk} and $s_{jk'}$. Thus, in our optimization problem, we assume function $F(\cdot)$ can be simplified as

$$F(r^a_{j*}, c^a_{j**'}) = F(c^a_{j**'}) = \frac{\sum_{k=1}^{|S_j|}\sum_{k'=1}^{S_j}\frac{f_{jkk'}}{c^d_{jkk'}}}{\sum_{k=1}^{|S_j|}\sum_{k'=1}^{S_j}\frac{f_{jkk'}}{c^a_{jkk'}} + \sum_{i=1}^{|M|}\sum_{k=1}^{|S_j|} l_{ijk}} \tag{7}$$

where $r^a_{j*} = r^d_{j*'}$, and $k \neq k'$.

Therefore, the SLA fulfillment of virtual network v_j is simplified as

$$Q_j = \begin{cases} 100\%, & \text{if } F(c^a_{j**'}) > 1 \\ F(c^a_{j**'}), & \text{otherwise.} \end{cases} \tag{8}$$

We consider the SLA fulfillment as a payoff function of each virtual network. We consider the virtual networks as game players since each virtual network behaves in a selfish way to maximize its utility. Thus, we formulate the optimization problem as a non-cooperative game between individual virtual networks, given by

$$\begin{aligned} &\max_{S_j} Q_j(L_j), \forall j \in [1, |V|]. \\ &s.t., \sum_{j=1}^{|V|}\sum_{k=1}^{|S_j|} r^a_{jk} \cdot l_{ijk} \leq R^c_i, \\ &\sum_{j=1}^{|V|}\sum_{k=1}^{|S_j|}\sum_{k'=1}^{|S_j|} c^a_{jkk'} l_{ijk} l_{i'jk'} \leq c_{ii'}, i \neq i', k \neq k'. \end{aligned} \tag{9}$$

In a non-cooperative game, a Pareto efficient solution is considered as an efficient manner. Thus, in the QoS-awared combined optimization, we want to find a Pareto efficient solution to optimize the SLA fulfillment of each virtual network.

We first define a Pareto efficient assignment in the virtual network optimization problem. Let vector $y = (L_1, ..., L_{|M|})$ denote the outcome of the game regarding the assignment of instance placement solutions of all the virtual networks, where L_j is the set of the placement for network v_j.

Definition 1: An assignment matrix y^*, is Pareto efficiency, if there exists no other assignment matrix y such that $Q_j(y) \geq Q_j(y^*)$ for all $j \in |V|$ and $Q_j(y) > Q_j(y^*)$ for some $j \in |V|$.

The problem of virtual network optimization problem: given a set of physical servers with a connected network and a set of virtual networks with instances, the virtual network optimization problem attempts to assign the network bandwidth to virtual networks and place instances to the physical servers such that the SLA-fulfillment of all virtual networks satisfies Definition 1.

Algorithm 2 Instance placement optimization for virtual networks

```
sort set M and matrix C in which c_12 ≥ c_13 ≥ ... ≥ c_21 ≥ c_23 ≥ ... ≥ c_|C|-1|C|;
∀i ∈ [1,|M|], j ∈ [1,|V|], k ∈ [1,|S_j|], l_ijk ← 0;
for j ← 1 to |V| do
    Sort S_j in which (Σ_{k'←1}^{|S_j|} c^d_{j1k'} + Σ_{k←1}^{|S_j|} c^d_{jk1}) ≥ (Σ_{k'←1}^{|S_j|} c^d_{j2k'} + Σ_{k←1}^{|S_j|} c^d_{jk2}) ≥ ... ≥ (Σ_{k'←1}^{|S_j|} c^d_{j|S_j|-1k'} Σ_{k←1}^{|S_j|} c^d_{jk|S_j|-1});
    for i ← 1 to |M| do
        for k ← 1 to S_j do
            if R^c_i ≥ r^d_jk and Q'_j > Q_j then
                place s_jk to m_i;
                l_ijk ← 1
                R^c_i ← R^c_i − r^c_jk;
            end if
        end for
    end for
    for k ← 1 to |S_j|−1 do
        for k' ← j + 1 to |S_j| do
            i ← arg place s_jk;
            i' ← arg place s_jk';
            if i ≠ i' then
                c_ii' ← c_ii' − c^d_jkk';
                c_i'i ← c_i'i − c^d_jk'k;
            end if
        end for
    end for
    sort set M and matrix C in which c_12 ≥ c_13 ≥ ... ≥ c_21 ≥ c_23 ≥ ... ≥ c_|C|-1|C|;
end for
```

5.2 Solving Virtual Network Optimization Problem

We design an algorithm shown in Algorithm 2 to find a Pareto efficient solution in the virtual network optimization problem. The algorithm first sorts the set M and matrix C in descending order by the available network bandwidth of each link. We also set the original placement matrix $L_j, j \in [1, |V|]$ equal 0. Then, the algorithm starts to traverse the ordered set $|V|$ from v_1 to $v_{|V|}$. In the traversal, the algorithm sorts the instances in descending order by the sum of required network bandwidth. Then, the algorithm places the ordered instances into ordered physical servers iteratively, if the resource capacity of the server is enough. After placement, the algorithm decreases the available bandwidth between the servers in which instances are placed. After placing all instances in one virtual network, set M and matrix C are sorted again with the descending order of the available network bandwidth of each link.

Lemma 1: The output vector $y^* = (L_1, L_2, ..., L_{|V|})$ of Algorithm 2 is a Pareto efficient solution.

Proof: We first assume a different assignment vector $y' = (L'_1, L'_2, ..., L'_{|V|})$ such that

$$\forall j \in [1, |V|] : Q_j(y') \geq Q_j(y^*), \text{ and} \tag{10}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2871118, IEEE Transactions on Cloud Computing

$$\exists j \in [1, |V|] : Q_j(y') > Q_j(y^*). \tag{11}$$

Let V_d denote the virtual networks having different placement between y' and $y*$. For $v_j \in V_d$, we first consider a case as

$$\begin{cases} l_{ijk} = 1, \\ l'_{ijk} = 0, \\ l_{i'jk} = 0, \\ l_{i'jk} = 1 \end{cases} \tag{12}$$

where the $l_{ijk} = 1$, instance s_{jk} is placed. Since the $l'_{ijk} = 1$, the s'_{jk} is also placed.

Then, we consider another case as

$$\begin{cases} l_{ijk} = 0, \\ l'_{ijk} = 1, \\ l_{i'jk} = 1, \\ l_{i'jk} = 0 \end{cases} \tag{13}$$

where both instance s_{jk} and s'_{jk} are placed.

Since we assume the algorithm can place all instances, $\forall k \in [1, |S_j|] : \exists i \in [1, |M|] : l_{ijk} = 1$. Thus, the third case is as

$$\begin{cases} l_{ijk} = 1, \\ l'_{ijk} = 0 \end{cases} \tag{14}$$

where $\forall i' \in [1, |M|] : l_{i'jk} \neq 1$.

In this case, because an instance is not placed in v_j with solution y', payoff $Q_j(y')$ is less than $Q_j(y^*)$. Therefore, from three cases, $Q_j(y') \leq Q_j(y^*)$, which is inconsistent with the assumed equation (10) and (11). As a result, the output y^* is a Pareto efficient solution. □

6 Performance Evaluation

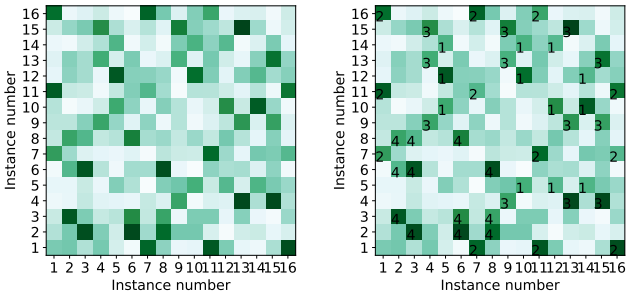
In this section, we first introduce the experiment settings and then discuss the experimental results of virtual network recognition and optimization.

6.1 Simulation Settings

We use a workstation computer as the simulation platform which is equipped a Core™ i7 4770 (8 MB Cache, up to 3.90 GHz) CPU, 16 GByte RAM, and 2 Tbyte HDD. We test each simulation 20 times and record the average results.

We first build a small cloud datacenter network on mininet 2.3.0 which is a populate emulation platform for SDN applications. We use a tree topology with 16 instances and 5 switches and set 4 leaf switches as 4 physic servers. The bandwidth between physical servers and the core switch is 40 Gbps. We implement a traffic monitor model in the Floodlight 1.2 controller then the controller deploys flow tables into each aggregation switch to monitor the end-to-end traffic information.

For large-scale experiments, we implement all simulations with Python 2.7.12 and NetworkX 1.11. We set the number of virtual servers as 100. We use a three



(a) Network traffic heat map (b) Result

Fig. 3. Example of Virtual network recognition algorithm

layers tree topology network to connect all servers. The bandwidth between the aggregation switch and instances is set to 10 Gbps and the bandwidth between the aggregation switch and core switch is set to 40 Gbps, which comes from a typical datacenter settings. The maximum number of user virtual networks is set to 50, and the number of instances per each virtual network is uniform distributed from 5 to 15.

In the simulations, we choose two types of workloads for measuring our method, one type is the big data processing which has heavy traffic and large virtual networks, and another is the web service which has lighter traffic and small virtual networks.

For the traffic information from big data computing applications, we record the traffic datasets by a virtual cluster deployed in Google Compute Engine, which has maximum 30 instances. We install an Apache Hadoop and a Spark system in the cluster and run several example programs with a different number of instances to generate traffic datasets. We choose Apache Hadoop 2.7.3 and Spark 2.0 as the big data applications. For the traffic information from web services, we use a traffic trace dataset from a real-world IaaS environment [39].

We define an integer number for measuring the resources of each server, and the resource capacity of each physical server is 100. The required resource of an instance in each virtual network is uniform distributed from 3 to 30 and the required bandwidth is set to 1 Gbps which is a common value in an IaaS platform.

For different number of instances in the virtual network, we set the traffic from the recorded datasets with the same size. We randomly place the instances of each virtual network to virtual servers and replay the traffic dataset in each link.

6.2 Recognition Accuracy

We first test the virtual network recognition in the small cloud datacenter network with the monitor module in the Floodlight controller. As shown in 3, we use a network traffic heat map to show the settings and result. The traffic between instances is shown as a heat map in Fig. 3(a). The deeper color means heavier traffic while lighter color means light traffic. We execute our virtual

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2871118, IEEE Transactions on Cloud Computing

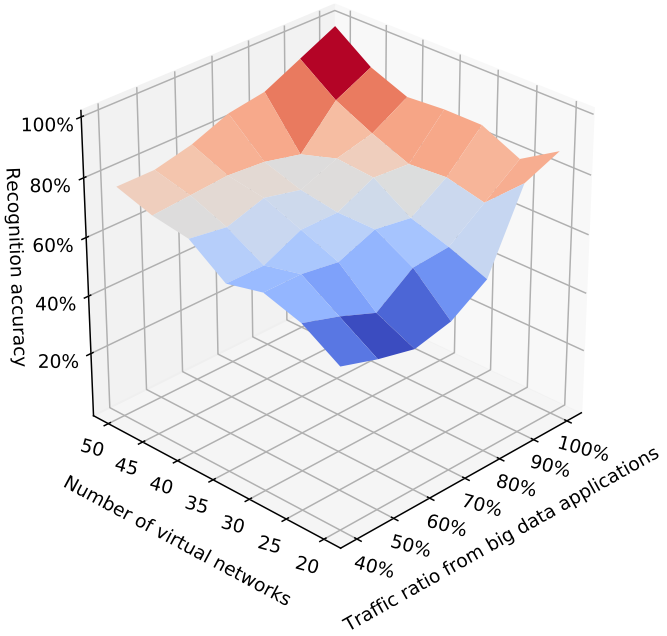


Fig. 4. Accuracy of virtual network recognition

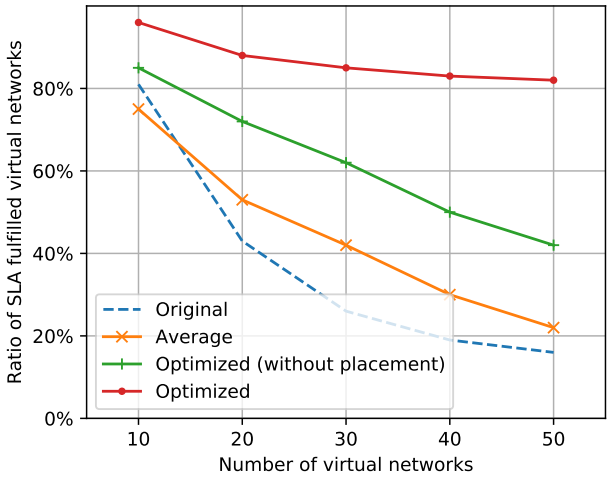


Fig. 5. Ratio of SLA fulfilled virtual networks with different number of virtual networks

network recognition algorithm, and the result is shown in Fig. 3(b). In the same heat map, we put a numerical digit on the traffic to mark which virtual network the traffic belongs to and use 1 to 4 to mark 4 different virtual networks. The recognition algorithm works well and all virtual networks are recognized correctly.

Then, we test the recognition accuracy in large-scale simulations. We adjust the ratio of big data traffic to total traffic in simulations. Since users will deploy other applications in cloud, different traffic records will influence the accuracy of the recognition algorithm. In the simulation, we set the ratio of big data computing traffic to total traffic from 100% to 20%. We also set the number of virtual networks from 20 to 50 and increase 5 instances in each step. Then, we execute the recognition algorithm to find the virtual networks from the traffic records. After recognition, we compare the instances in recognized virtual networks and original virtual networks. If the original virtual network and the recognized one have same instances, we consider that recognized virtual network is correct. The accuracy is the ratio of the number of correct virtual networks to the number of all virtual networks.

From Fig. 4, our recognition algorithm recognizes virtual networks with an acceptable accuracy. When the traffic ratio of big data computing is more than 80%, the accuracy is near to 70% with 30 virtual networks. The accuracy decreases with fewer instances and lower traffic ratio from big data computing. When the traffic ratio from big data computing decreases to 40%, the recognition accuracy decreases to less than 80% with 50 virtual networks or 40% with 20 virtual networks. Thus, both the traffic type and the number of instances influence the recognition accuracy, which means our recognition method is suitable for heavy traffic scenarios.

6.3 QoS Performance

The QoS performance is measured by the SLA fulfillment ratio. In simulations, we test the ratio of the number of SLA fulfilled virtual networks to the number of all virtual networks. We execute the recognition algorithm first and then run the optimization algorithm with the recognized virtual networks. The ratio of big data traffic is set to 80%. We also compare the performance of our method to original settings and average bandwidth assignment which averagely assigns bandwidth of each link to instances. Meanwhile, for studying the advantage of instance placement, we also test the performance of our virtual network optimization solution without instance placement.

We first test the QoS performance with a different number of virtual networks. We increase the number of virtual networks from 10 to 50 and increase 10 virtual networks in each step. As shown in Fig. 5, the SLA fulfilled ratio decreases with more virtual networks. When the number of virtual networks is set to 10, the SLA fulfilled ratio is more than 95% after optimization and near to 80% by original settings. Without instance placement, the performance of our optimization method is near to the original settings. When the number of virtual networks increases to 50, SLA fulfilled ratio with original settings is less than 20%, while with our optimization method with instance placement, more than 80% virtual networks are SLA fulfilled with instance placement. Thus, instance placement increases the efficiency of the virtual network optimization obviously with more virtual networks. Average bandwidth assignment performs better than original settings, when the number of virtual networks increases to 20.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2018.2871118, IEEE Transactions on Cloud Computing

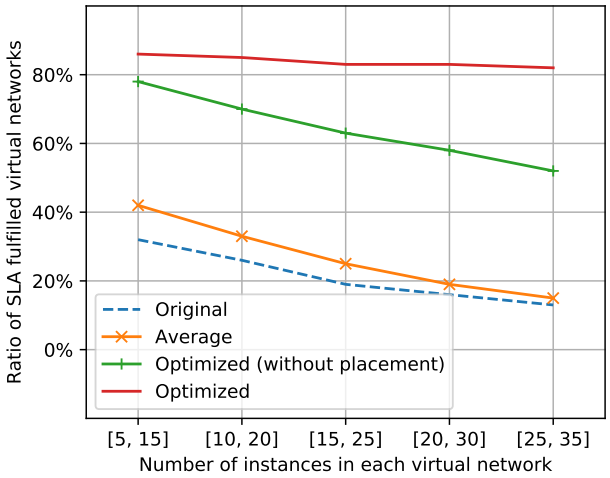


Fig. 6. Ratio of SLA fulfulled virtual networks with different number of instances in each virtual network

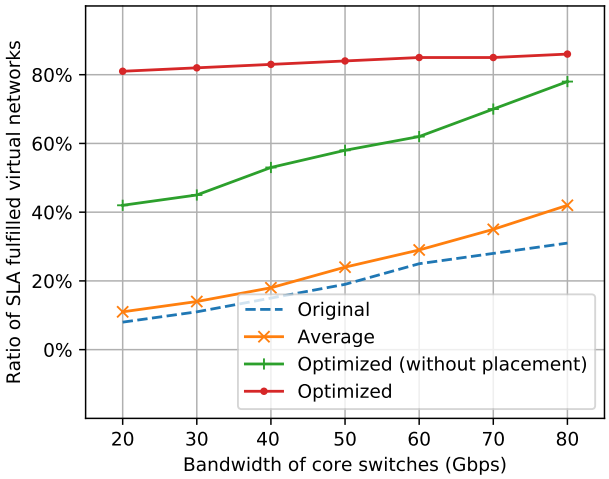


Fig. 7. Ratio of SLA fulfilled virtual networks with different bandwidth betwee aggreation and core switches

We test the QoS performance with different number of instances in each virtual networks. We set the number of virtual networks as 50, increase the number of instances in each virtual network from [5, 15] to [25, 35], and increase 10 instances in each step. As shown in Fig.6, the SLA fulfilled ratio is decreased with more instances per each virtual network. When the number of instances in each virtual network is uniform distributed from 5 to 15, the SLA fulfilled ratio is more than 30% with original settings and more than 80% with our optimization. Our optimization method without instance placement performs similarly to the one with placement, while average assignment performs better than original settings. When we uniform distribute the number of instances per each virtual network from 25 to 35, the SLA fulfilled ratio is less than 20% with original instance placement and more than 80% after our optimization. The ratio of SLA fulfilled virtual networks is near to 50 % by our optimization without instance placement, while the performance of average assignment is near to the original settings. Thus, instance placement improves efficiency of the optimization algorithm with more instances in virtual networks.

We also test the influence on the QoS performance with different bandwidth of core switches. The number of instances per each virtual network is uniform distributed from 5 to 35. The bandwidth of core switches increases from 20 Gbps to 80 Gbps and increases 10Gbps in each step. As shown in Fig. 7, SLA fulfilled ratio increases with higher bandwidth of core switches. When the bandwidth of core switches is set to 20 Gbps, SLA fulfilled ratio is less than 10% with original settings and near to 80% with our optimization. The fulfilled ratio with our optimization method without instance placement is near to 40%. Average assignment performs similarly with original settings. After the bandwidth of core switches increases to 80Gbps, the ratio of fulfilled virtual networks increases to 30% with original instance placement. The fulfilled ratio with our optimization without instance placement is increased to 80% because of the enough bandwidth of core switches. Therefore, the performance is not improved obviously with enough bandwidth of core switches.

As a result, our virtual network recognition and optimization method performs better than the average assignment and the original settings. The performance of our solution is slightly influenced by different virtual network parameters. In all simulation results, our optimization method with instance placement achieves a SLA fulfilled ratio of more than 80%. Meanwhile, the performance with instance placement is related to the available network resource. Instance placement improves the number of SLA fulfilled virtual networks with the scarce network resource. When the cloud environment provides enough network resource, bandwidth adjustment without instance placement can also improve the SLA fulfilled ratio.

7 Conclusion and Future Work

In this paper, we first investigate the virtual network recognition problem in a general-purpose cloud environment. Since the user virtual network topologies is not acknowledged, it is hard to implement an efficient optimization for cloud computing. Thus, we propose an optimization method that first recognizes virtual networks then optimizes the performance of the recognized virtual networks. We design a virtual network recognition algorithm by formulating the recognition problem as a classic community detection problem. The optimization algorithm focuses on the QoS performance of each virtual network. We adopt the SLA fulfillment as the measurement of QoS and formulate the optimization problem as a non-cooperative game with a Pareto

efficient solution. Finally, we evaluate our solution and compare the performance to the original placement. In the future, we first plan to adopt pattern recognition to detect different applications from the network traffic records, since different applications have different requirements on cloud resources.

ACKNOWLEDGMENTS

This work is supported by JSPS KAKENHI Grant Number JP16K00117, JP15K15976, JP17K12669, KDDI Foundation, and Research Fund for Postdoctoral Program of Muroran Institute of Technology. Mianxiong Dong is the corresponding author.

REFERENCES

[1] B. He, M. Lu, K. Yang, R. Fang, N. K. Govindaraju, Q. Luo, and P. V. Sander, "Relational query coprocessing on graphics processors," *ACM Trans. Database Syst.*, vol. 34, no. 4, pp. 21:1–21:39, Dec. 2009.

[2] L. Hu, X. Che, and Z. Xie, "Gpgpu cloud: A paradigm for general purpose computing," *Tsinghua Science and Technology*, vol. 18, no. 1, pp. 22–23, Feb 2013.

[3] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, ser. CCGRID '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 826–831.

[4] 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.

[5] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, April 2015, pp. 1–9.

[6] T. Truong Huu, G. Koslovski, F. Anhalt, J. Montagnat, and P. Vicat-Blanc Primet, "Joint elastic cloud and virtual network framework for application performance-cost optimization," *Journal of Grid Computing*, vol. 9, no. 1, pp. 27–47, 2011.

[7] M. Xu, L. Cui, H. Wang, and Y. Bi, "A multiple qos constrained scheduling strategy of multiple workflows for cloud computing," in *2009 IEEE International Symposium on Parallel and Distributed Processing with Applications*, Aug 2009, pp. 629–634.

[8] R. L. Krutz and R. D. Vines, *Cloud security: A comprehensive guide to secure cloud computing*. Wiley Publishing, 2010.

[9] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, November 2013.

[10] P. Wieder, J. M. Butler, W. Theilmann, and R. Yahyapour, *Service level agreements for cloud computing*. Springer Science & Business Media, 2011.

[11] L. Wang and E. Gelenbe, "Adaptive dispatching of tasks in the cloud," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[12] L. Wang, O. Brun, and E. Gelenbe, "Adaptive workload distribution for local and remote clouds," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct 2016, pp. 003 984–003 988.

[13] E. Gelenbe and L. Wang, "Tap: A task allocation platform for the eu fp7 panacea project," in *Advances in Service-Oriented and Cloud Computing: Workshops of ESOCC 2015, Taormina, Italy, September 15–17, 2015, Revised Selected Papers*, vol. 567, 2016, p. 425.

[14] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 2, pp. 909–928, Second 2013.

[15] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, "Understanding tcp incast throughput collapse in datacenter networks," in *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, ser. WREN '09. New York, NY, USA: ACM, 2009, pp. 73–82.

[16] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of the 29th Conference on Information Communications*, ser. INFOCOM'10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 1154–1162.

[17] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, and E. Snible, "Improving performance and availability of services hosted on iaas clouds with structural constraint-aware virtual machine placement," in *2011 IEEE International Conference on Services Computing*, July 2011, pp. 72–79.

[18] V. Shrivastava, P. Zerfos, K. w. Lee, H. Jamjoom, Y. H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *2011 Proceedings IEEE INFOCOM*, April 2011, pp. 66–70.

[19] X. Wen, K. Chen, Y. Chen, Y. Liu, Y. Xia, and C. Hu, "Virtualknotter: Online virtual machine shuffling for congestion resolving in virtualized datacenter," in *2012 IEEE 32nd International Conference on Distributed Computing Systems*, June 2012, pp. 12–21.

[20] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera, "A stable network-aware vm placement for cloud systems," in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, May 2012, pp. 498–506.

[21] M. Alicherry and T. V. Lakshman, "Network aware resource allocation in distributed clouds," in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 963–971.

[22] L. Hu, K. D. Ryu, D. D. Silva, and K. Schwan, "v-bundle: Flexible group resource offerings in clouds," in *2012 IEEE 32nd International Conference on Distributed Computing Systems*, June 2012, pp. 406–415.

[23] T. Benson, A. Akella, A. Shaikh, and S. Sahu, "Cloudnaas: A cloud networking platform for enterprise applications," in *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, ser. SOCC '11. New York, NY, USA: ACM, 2011, pp. 8:1–8:13.

[24] F. Francois and E. Gelenbe, "Towards a cognitive routing engine for software defined networks," in *2016 IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6.

[25] F. Francois and E. Gelenbe, "Optimizing secure sdn-enabled inter-data centre overlay networks through cognitive routing," in *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Sept 2016, pp. 283–288.

[26] A. Tavakoli, M. Casado, T. Koponen, and S. Shenker, "Applying nox to the datacenter," in *Proc. of workshop on Hot Topics in Networks (HotNets-VIII)*, 2009.

[27] B. Pfaff, J. Pettit, K. Amidon, M. Casado, T. Koponen, and S. Shenker, "Extending networking into the virtualization layer," in *HotNets*, 2009.

[28] B. P. M. C. Justin Pettit, Jesse Gross, "Virtual switching in an era of advanced edges," in *2nd Workshop on Data Center-Converged and Virtual Ethernet Switching*, 2010, pp. 1–7.

[29] M. Moshref, M. Yu, A. Sharma, and R. Govindan, "Scalable rule management for data centers," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, ser. nsdi'13. Berkeley, CA, USA: USENIX Association, 2013, pp. 157–170.

[30] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.

[31] Q. Li, J. Huai, J. Li, T. Wo, and M. Wen, "Hypermip: Hypervisor controlled mobile ip for virtual machine live migration across networks," in *2008 11th IEEE High Assurance Systems Engineering Symposium*, Dec 2008, pp. 80–88.

[32] P. Raad, G. Colombo, D. P. Chi, S. Secci, A. Cianfrani, P. Gallard, and G. Pujolle, "Achieving sub-second downtimes in internet-wide virtual machine live migrations in lisp networks," in *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, May 2013, pp. 286–293.

[33] M. Coudron, S. Secci, G. Maier, G. Pujolle, and A. Pattavina, "Boosting cloud communications through a crosslayer multipath protocol architecture," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Nov 2013, pp. 1–8.

[34] The OpenDaylight Project, Inc., "OpenDaylight - Technical Overview," 2013. [Online]. Available: http://www.opendaylight.org/project/technical-overview

[35] P. S. Pisa, N. C. Fernandes, H. E. T. Carvalho, M. D. D. Moreira, M. E. M. Campista, L. H. M. K. Costa, and O. C. M. B. Duarte,

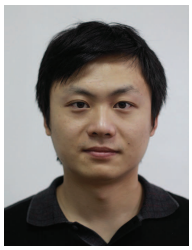
OpenFlow and Xen-Based Virtual Network Migration. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 170–181.

[36] V. Mann, A. Vishnoi, K. Kannan, and S. Kalyanaraman, "Crossroads: Seamless vm mobility across data centers through software defined networking," in *2012 IEEE Network Operations and Management Symposium*, April 2012, pp. 88–96.

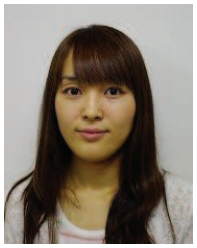
[37] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3C5, pp. 75 – 174, 2010.

[38] J. Leskovec, K. J. Lang, and M. Mahoney, "Empirical comparison of algorithms for network community detection," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: ACM, 2010, pp. 631–640.

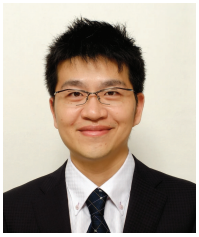
[39] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 267–280.



He Li received the B.S., M.S. degrees in Computer Science and Engineering from Huazhong University of Science and Technology in 2007 and 2009, respectively, and the Ph.D. degree in Computer Science and Engineering from The University of Aizu in 2015. He is currently a Postdoctoral Fellow with Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan. His research interests include cloud computing and software defined networking. He has received the best paper award from IEEE VTC2016-Fall. Dr. Li serves as an Associate Editor for Human-centric Computing and Information Sciences (HCIS), as well as a Guest Associate Editor for IEICE Transactions on Information and Systems. He is the recipient of IEEE TCSC Outstanding Ph.D. Dissertation Award 2016.



Kaoru Ota was born in Aizu-Wakamatsu, Japan. She received M.S. degree in Computer Science from Oklahoma State University, USA in 2008, B.S. and Ph.D. degrees in Computer Science and Engineering from The University of Aizu, Japan in 2006, 2012, respectively. She is currently an Assistant Professor with Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan. From March 2010 to March 2011, she was a visiting scholar at University of Waterloo, Canada. Also she was a Japan Society of the Promotion of Science (JSPS) research fellow with Kato-Nishiyama Lab at Graduate School of Information Sciences at Tohoku University, Japan from April 2012 to April 2013. Her research interests include Wireless Networks, Cloud Computing, and Cyber-physical Systems. Dr. Ota has received best paper awards from ICA3PP 2014, GPC 2015, IEEE DASC 2015, IEEE VTC 2016-Fall, FCST 2017, 2017 IET Communications Premium Award and IEEE ComSoc CSIM Best Conference Paper Award 2018. She is an editor of IEEE Transactions on Vehicular Technology (TVT), IEEE Communications Letters, Peer-to-Peer Networking and Applications (Springer), Ad Hoc & Sensor Wireless Networks, International Journal of Embedded Systems (Inderscience) and Smart Technologies for Emergency Response & Disaster Management (IGI Global), as well as a guest editor of ACM Transactions on Multimedia Computing, Communications and Applications (leading), IEEE Internet of Things Journal, IEEE Communications Magazine, IEEE Network, IEEE Wireless Communications, IEEE Access, IEICE Transactions on Information and Systems, and Ad Hoc & Sensor Wireless Networks (Old City Publishing). She is the recipient of IEEE TCSC Early Career Award 2017.



Mianxiong Dong received B.S., M.S. and Ph.D. in Computer Science and Engineering from The University of Aizu, Japan. He is currently an Associate Professor in the Department of Information and Electronic Engineering at the Muroran Institute of Technology, Japan. He was a JSPS Research Fellow with School of Computer Science and Engineering, The University of Aizu, Japan and was a visiting scholar with BBCR group at University of Waterloo, Canada supported by JSPS Excellent Young Researcher Overseas Visit Program from April 2010 to August 2011. Dr. Dong was selected as a Foreigner Research Fellow (a total of 3 recipients all over Japan) by NEC C&C Foundation in 2011. His research interests include Wireless Networks, Cloud Computing, and Cyber-physical Systems. He has received best paper awards from IEEE HPCC 2008, IEEE ICESS 2008, ICA3PP 2014, GPC 2015, IEEE DASC 2015, IEEE VTC 2016-Fall, FCST 2017, 2017 IET Communications Premium Award and IEEE ComSoc CSIM Best Conference Paper Award 2018. Dr. Dong serves as an Editor for IEEE Transactions on Green Communications and Networking (TGCN), IEEE Communications Surveys and Tutorials, IEEE Network, IEEE Wireless Communications Letters, IEEE Cloud Computing, IEEE Access, as well as a leading guest editor for ACM Transactions on Multimedia Computing, Communications and Applications (TOMM), IEEE Transactions on Emerging Topics in Computing (TETC), IEEE Transactions on Computational Social Systems (TCSS). He has been serving as the Vice Chair of IEEE Communications Society Asia/Pacific Region Information Services Committee and Meetings and Conference Committee, Leading Symposium Chair of IEEE ICC 2019, Student Travel Grants Chair of IEEE GLOBECOM 2019, and Symposium Chair of IEEE GLOBECOM 2016, 2017. He is the recipient of IEEE TCSC Early Career Award 2016, IEEE SCSTC Outstanding Young Researcher Award 2017, The 12th IEEE ComSoc Asia-Pacific Young Researcher Award 2017 and Funai Research Award 2018. He is currently the Member of Board of Governors and Chair of Student Fellowship Committee of IEEE Vehicular Technology Society.