Efficient Embedding of Virtual Networks to Distributed Clouds Via Exploring Periodic Resource Demands

Zichuan Xu, *Student Member, IEEE,* Weifa Liang, *Senior Member, IEEE,* and Qiufen Xia, *Student Member, IEEE*

Abstract—Cloud computing built on virtualization technologies promises provisioning elastic computing and bandwidth resource services for enterprises that outsource their IT services as virtual networks. To share the cloud resources efficiently among different enterprise IT services, embedding their virtual networks into a distributed cloud that consists of multiple data centers, poses great challenges. Motivated by the fact that most virtual networks operate on long-term basis and have the characteristics of periodic resource demands, in this paper we study the virtual network embedding problem of embedding as many virtual networks as possible to a distributed cloud such that the revenue collected by the cloud service provider is maximized, while the Service Level Agreements (SLAs) between enterprises and the cloud service provider are met. We first propose an efficient embedding algorithm for the problem, by incorporating a novel embedding metric that accurately models the dynamic workloads on both data centers and inter-data center links, provided that the periodic resource demands of each virtual network are given and all virtual networks have identical resource demand periods. We then show how to extend this algorithm for the problem when different virtual networks may have different resource demand periods. Furthermore, we also develop a prediction mechanism to predict the periodic resource demands of each virtual network if its resource demands are not given in advance. We finally evaluate the performance of the proposed algorithms through experimental simulation based on both synthetic and real network topologies. Experimental results demonstrate that the proposed algorithms outperform existing algorithms from 10% to 31% in terms of performance improvement.

Index Terms—Virtual network embedding; cloud resource provisioning; embedding algorithms; periodic resource demands; distributed clouds; cloud computing.

1 Introduction

Enterprises nowadays are embracing a new computing paradigm by outsourcing their IT service networks as virtual networks to clouds for cost savings. For example, a company operating video conferencing service could run on a virtual network with a stringent quality of service (QoS) requirement, whereas a university delivering online courses for distance education service may run a virtual network for real-time online course delivery. These two different virtual networks can be accommodated by a distributed cloud, which is also referred to as *the substrate network* that consists of multiple data centers connected through high-speed optical links [35], [36], [37]. A fundamental problem for this substrate network is how to efficiently embed as many virtual networks as possible to it such that the revenue of the cloud service provider is maximized, while the Service Level Agreements (SLAs) between users and the cloud service provider are met. This problem is referred to as the *Virtual Network Embedding* (VNE) problem, which has been extensively studied in the past few years [3], [9], [10], [16], [23], [38], [39], [44].

- *Z. Xu, W. Liang and Q. Xia are with the Research School of Computer Science, The Australian National University, Canberra, ACT 0200, Australia. E-mails: edward.xu@anu.edu.au, wliang@cs.anu.edu.au, and qiufen.xia@anu.edu.au*

Most existing studies on the VNE problem in literature focused on resource provisions by reserving the maximum resource demands for a virtual network throughout its whole lifetime [8], [10], [39], [44], [42]. Such a conservative resource provision scheme however causes up to 85 percentage of the cloud resources under-utilized in most time, resulting in enormous resource wastage and economic loss [32], [35], [36]. Fortunately, nearly 90% of enterprise IT services exhibit the characteristics of periodic resource demands [18]. By making use of this periodic resource demand property, the utilization ratios of cloud resources can be substantially improved if the demanded resources by different virtual networks can be shared through exploring their resource demand periods, which can be illustrated by an example in Fig. 1, where a virtual network A providing office users with virtual desktop services usually experiences low-workloads in weekends, whereas another virtual network B hosting online gaming services has high-workloads in weekends. If embedding A and B to the substrate network while meeting their individual maximum demands, only one of them can be embedded. However, they both can be embedded concurrently if their time-varying resource demands are complementary. This could potentially reduce the operational cost of the cloud service provider.

Due to the heterogeneity of substrate resources 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.2016.2535215, IEEE Transactions on Cloud Computing

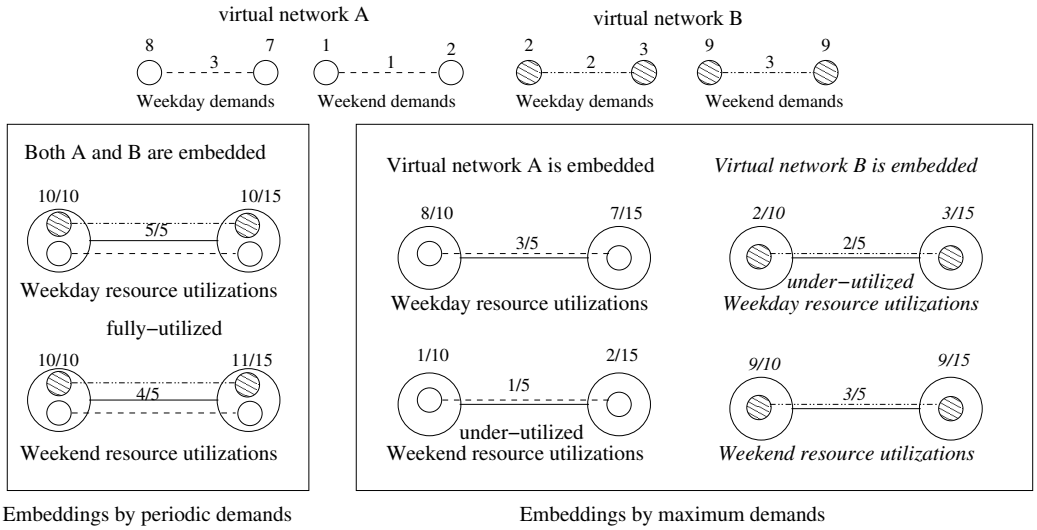


Fig. 1: A motivated example of resource sharing among virtual networks with periodic resource demands

without the knowledge of periodic resource demands of each virtual network in advance, it poses a great challenge to embed as many virtual networks as possible into a substrate network such that the revenue collected by the cloud service provider is maximized, while meeting the SLAs of users. To efficiently embed different virtual networks to a distributed cloud with dynamic workloads on its data centers and links, an embedding metric that can accurately model the dynamic workloads of cloud resources needs to be developed, and an efficient embedding method that embeds each virtual network to the substrate network is required to be devised such that both the computing and the bandwidth demands of the virtual network are met. In this paper, we will introduce a novel embedding metric for virtual network embedding, and develop efficient algorithms for virtual network embedding by exploring the periodic resource demands of virtual networks, based on the proposed embedding metric.

Although the embedding of a single virtual network to the substrate network have been intensively studied in recent years, to the best of our knowledge, we have not been aware of any study of embedding multiple virtual networks with periodic resource demands to a distributed cloud simultaneously. We are the first to consider the embedding of multiple virtual networks with periodic resource demands while meeting the SLAs of different users, through introducing a novel embedding metric that can accurately capture the dynamic workloads of computing and bandwidth resources in a distributed cloud.

The main contributions of this paper are as follows. We first propose an embedding algorithm for the VNE problem, by employing a novel metric to capture the workloads on substrate nodes and substrate links, assuming that all virtual networks have identical resource demand periods and their resource demands are given in advance. We then extend the proposed embedding algorithm for a general setting of the problem where different virtual networks may have different resource demands periods, or such resource demand periods may not be given in advance, by developing a resource demand prediction algorithm for each virtual network. We finally evaluate the performance of the proposed algorithms through experimental simulations, based on both synthetic and real substrate network topologies. Experimental results show that the proposed algorithms outperform existing algorithms, improving the revenue of the cloud service provider from 10% to 31%.

The remainder of the paper is organized as follows. Section 2 introduces related work, followed by the system model and problem definitions in Section 3. Sections 4 and 5 propose VNE algorithms with and without the periodic resource demands of virtual networks. Section 6 evaluates the performance of the proposed algorithms through experimental simulations. The conclusion is given in Section 7.

2 Related Work

Network virtualization has been recognized as a promising solution to the perceived ossification of the current Internet [1], [14], which can improve performance of inter- and intra-data center networks [2], [3], [4], [9], [11], [20], [21], [29], [33], [34]. For example, Guo *et. al.* [20] proposed a bandwidth reservation method between every pair of virtual machines (VMs) within a single data center. Ballani *et. al.* [2] proposed a virtual cluster model, in which all VMs are connected to a virtual switch to which an amount of bandwidth resource is allocated. Wood *et. al.* [33] aimed to enhance seamless interconnections of applications distributed in multiple data centers, by utilizing the Virtual Private Network (VPN) technique [15]. They however did not consider joint allocations of computing and network bandwidth resources. Their approaches thus are inapplicable to resource allocations for applications that rely on both computing and network bandwidth resources.

With the advancement of cloud computing and network virtualization technologies, joint allocation of net-

work bandwidth and computing resources becomes feasible, by utilizing virtual network embedding (VNE) technique. In general, most existing solutions to the VNE problem can be classified into two categories: *static* and *dynamic* resource provisioning. Static resource provisioning assumes that the resource demands of each virtual network do not change during the lifetime of the virtual network, whereas dynamic resource provisioning deals with the embedding of virtual networks with dynamic resource demands over time. Existing studies in literature focused mainly on static resource provisions [6], [8], [10], [12], [13], [26], [30], [39], [42], [44]. For example, Zhu *et. al.* [44] proposed a VNE algorithm for balancing the workload on both computing nodes and communication links, by introducing a node/link-stress concept, and jointly considered the workload on each node and the links incident to the node. Chowdhury *et. al.* [10] devised an algorithm that coordinates node and link mapping, by reducing the VNE problem to a multi-commodity flow problem under the constraint that each virtual node can only be mapped to a specific set of candidate substrate nodes. Cheng *et. al.* [8] proposed an embedding algorithm that is similar to the Google's PageRank algorithm, where both substrate and virtual nodes are ranked by their resource availabilities and the quality of link connections. Lischka *et. al.* [26] devised an online VNE algorithm by making use of the subgraph isomorphism detection with the aim of maximizing the revenue-to-cost ratio, where the revenue is the total amount of virtual resources requested by virtual networks, and the cost is the total amount of substrate resources spent in accommodating the virtual networks. There are other static approaches that have less constraints on nodes and links, e.g., splittable path routing [39], embedding a virtual node onto several substrate nodes [42], embedding a virtual network across different substrate networks [22], distributed and automatic embedding [23], or avoiding cloud resource fragmentation [13]. For example, Chiang *et. al.* [39] initialized the study of the VNE problem with splittable path routing, by embedding the traffic on each virtual link to multiple substrate paths in the substrate network. Houidi [22] treated virtual network providers as brokers by allowing a virtual network to be embedded to multiple substrate networks with the aim of reducing the embedding cost of infrastructure providers while increasing the acceptance ratio of user requests.

There are several studies focusing on dynamic resource provisioning, by reallocating under-utilized resources to other virtual network requests [7], [31], [40], [41], [43]. For example, Zhang *et. al.* [40] studied the VNE problem by considering opportunistic resource sharing and topology-aware node ranking. They assumed that each virtual network has a basic and maximum demands with certain probabilities. Such an assumption may not be realistic as it is very unlikely that a user can provide the detailed resource demands of its virtual network in advance. The other dynamic resource provisioning approaches however perform periodic reconfigurations/migrations of implemented virtual networks, which may not be feasible in practice, due to high migration costs and/or violations of the agreed SLA requirements [5]. For example, Houidi *et. al.* [24] proposed an adaptive VNE algorithm that dynamically identifies new candidate substrate resources to cater dynamic topologies and dynamic communication requirements of virtual networks. Similarly, there are approaches in [7], [43] dealing with evolving virtual networks in terms of topologies and resource demands. For example, Sun *et. al.* [31] devised virtual network migration algorithms to deal with evolving virtual networks. Zhang *et. al.* [41] studied a scenario where both the demands of virtual networks and the capacity of a substrate network change over time. Unlike these mentioned previous works, in this paper we deal with dynamic resource provisions for virtual networks, by exploring periodic resource demands of virtual networks. The essential differences between our work and existing ones lie in a novel embedding metric that can model the workloads of both substrate nodes and substrate links accurately over time, and the exploration of periodic resource demands of virtual networks.

3 PRELIMINARIES

In this section we first introduce the substrate and virtual networks. We then provide the revenue and cost models of virtual network embedding. We finally define the embedding problems of virtual networks precisely.

3.1 Substrate and virtual networks

A *substrate network* is represented by a node-and-edge weighted, undirected graph $G^s = (N^s, E^s)$, where N^s and E^s are the sets of substrate nodes and links, respectively. Denote by n^s a substrate node in N^s and e^s a substrate link in E^s. Each n^s represents a data center and each e^s denotes a communication link (or a path) between the two data centers corresponding to its two endpoints. Denote by $C(n^s)$ the capacity of computing resource in n^s and $B(e^s)$ the bandwidth capacity on e^s.

A *virtual network* can be represented by a node-and-edge weighted, undirected graph $G^v = (N^v, E^v)$, where N^v and E^v are the sets of virtual nodes and virtual links. Each virtual node $n^v \in N^v$ represents a set of virtual machines that host specific applications. Each virtual edge $e^v \in E^v$ represents a communication link between two virtual nodes. Denote by $C(n^v)$ and $B(e^v)$ the *maximum amounts of computing and bandwidth resource demands* by virtual node n^v and virtual link e^v in G^v, respectively. Recall that a substrate network represents a distributed cloud, while a virtual network represents an enterprise IT service network.

Assume that time is divided into equal *intervals*, and each interval is further divided into equal numbers of *time slots*. Let i be the current interval and T the number of time slots in each interval. We assume that each interval i corresponds to *one resource demand period*. We further assume that virtual network requests arrive into

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.2016.2535215, IEEE Transactions on Cloud Computing

the system one by one without the knowledge of future arrivals. All arrived requests between time slot $(i-1)$ and time slot i will be examined in the beginning of time slot i, i.e., whether a request arrived during this period will be admitted (embedded) by the system will be determined in the beginning of time slot i. We say a virtual network $G^v = (N^v, E^v)$ with *a duration* $\tau(G^v)$ in the granularity of weeks or months if it is embedded to the substrate network, it will stay there until its duration $\tau(G^v)$ expires. Fig. 2 gives an example of virtual network embedding.

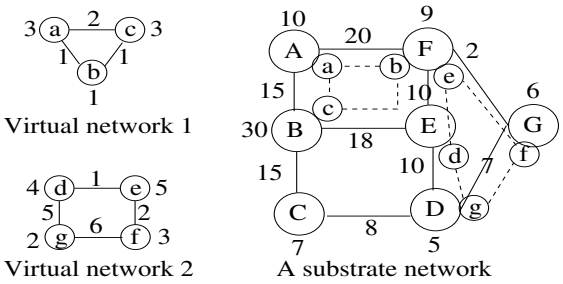


Fig. 2: Virtual network embedding

3.2 Periodic resource demands

Most enterprise IT services exhibit periodic resource demands [18]. For instance, an enterprise that provides university email services has weekly resource demands due to the weekly activity patterns of university users. Although periodic resource demands of a virtual network request typically are not known when the request initially arrived, they can be predicted by analyzing the resource demand history of the virtual network implementation, using offline profiling and online calibration [28].

Given an embedded virtual network $G^v(N^v, E^v)$, denote by $\hat{c}(n^v, i, t)$ and $\hat{b}(e^v, i, t)$ the predicted computing and bandwidth resource demands of a virtual node $n^v \in N^v$ and a virtual link $e^v \in E^v$ at time slot t in interval i. Let $c(n^v, i, t)$ and $b(e^v, i, t)$ be the actual amounts of demanded computing and bandwidth resources at virtual node n^v and virtual link e^v at time slot t in interval i. *The periodic computing and bandwidth resource demands* of G^v are defined as the resource demands that will be repeated in each interval, i.e., $c(n^v, i, t) = c(n^v, i', t)$ and $b(e^v, i, t) = b(e^v, i', t)$ for each time slot t at different intervals i and i' with $i \neq i'$. The amount of available resources of substrate network G^s can be derived from the accumulative resources allocated to all embedded virtual networks at time slot t in interval i. Denote by $P(n^s, i, t)$ and $P(e^s, i, t)$ the amounts of available computing and bandwidth resources in node n^s and link e^s of G^s at time slot t in interval i, where $1 \leq t \leq T$.

3.3 Revenue and cost models

The revenue collected by a cloud service provider by embedding virtual networks to the cloud can be defined differently, if different economic models are adopted. Similar to the revenue models in previous studies [10], [40], [44], in this paper the revenue collected by embedding a virtual network G^v to G^s is the sum of revenues of the usages of computing and bandwidth resources by G^v for its occupation period $\tau(G^v)$ in G^s, where c_c and c_b are the costs of unit computing and bandwidth resources, respectively. Denote by $\mathbb{R}(G^v)$ the revenue received by embedding G^v, then,

$$\mathbb{R}(G^v) = \Big(\sum_{n^v \in N^v} C(n^v) \cdot c_c + \sum_{e^v \in E^v} B(e^v) \cdot c_b \Big) \cdot \tau(G^v). \tag{1}$$

To provide the demanded computing and bandwidth resources to a virtual network $G^v = (N^v, E^v)$ while meeting its SLA, the cloud service provider consumes its resources such as electricity, software and hardware that incur the service (operational) cost. The service cost of embedding a virtual network G^v thus is defined as the sum of the usage costs of amounts of cloud resources within each time slot during its duration. Let $\mathbb{C}(G^v)$ be the service cost of an embedded virtual network G^v, then,

$$\mathbb{C}(G^v) = \sum_{i=1}^{\tau(G^v)} \sum_{t=1}^{T} \Big(\sum_{n^v \in N^v} c(n^v, i, t) \cdot c_c + \sum_{e^v \in E^v} \sum_{e^s \in E^s} l_{e^s}^{e^v} b(e^v, i, t) \cdot c_b \Big), \tag{2}$$

where $l_{e^s}^{e^v}$ is 1 if virtual link $e^v \in E^v$ is embedded to a path in G^s while $e^s \in E^s$ is a link in the path; 0 otherwise.

3.4 Problem definitions

Given a monitoring period consisting of I intervals with each having T equal time slots, assume that virtual network requests arrive one by one without the knowledge of future arrivals. The arrived requests will be scheduled in the beginning of each time slot. Let $\mathcal{G}(i,t)$ be the set of arrived virtual network requests in the beginning of time slot t in interval i, in which each virtual network G^v spans $\tau(G^v)$ intervals in the substrate network G^s. Each request exhibits periodic resource demands, i.e., it has the same amounts of computing and bandwidth resource demands in each interval i, $c(n^v, i, t) = c(n^v, i', t)$ and $b(e^v, i, t) = b(e^v, i', t)$ for any two intervals i and i' with $i \neq i'$ during its duration $\tau(G^v)$ in G^s and t is a time slot with $1 \leq t \leq T$.

The *virtual network embedding problem with the knowledge of periodic resource demands* is to embed as many virtual networks in $\mathcal{G}(i,t)$ as possible to the substrate network G^s for a given monitoring period I, such that the revenue of the cloud service provider of G^s is maximized, subject to the resource demands of each embedded virtual network at each time slot being met, where $1 \leq i \leq I$ and $1 \leq t \leq T$.

The *virtual network embedding problem without the knowledge of periodic resource demands* can be defined similarly, which is to embed as many virtual networks in $\mathcal{G}(i,t)$ as possible to the substrate network without the knowledge

TABLE 1: Symbols

Symbols	Notations
i and T	the current interval and the number of time slots of each interval
$G^s = (N^s, E^s)$	a substrate network with substrate node set N^s and substrate link set E^s
n^s and e^s	a substrate node in N^s and a substrate link in E^s
$C(n^s)$ and $B(e^s)$	the computing capacity of n^s and the bandwidth capacity of e^s
$P(n^s, i, t)$ and $P(e^s, i, t)$	the amounts of available computing and bandwidth resources in n^s and e^s at time slot t in interval i
$G^v = (N^v, E^v)$	a virtual network with virtual node set E^v and virtual link set E^v
n^v and e^v	a virtual node in N^v and a virtual link in E^v
$C(n^v)$ and $B(e^v)$	the maximum amounts of computing and network bandwidth resources demanded by n^v and e^v
$\hat{c}(n^v, i, t)$ and $\hat{b}(e^v, i, t)$	the predicted amounts of computing and bandwidth resource demands of n^v and e^v at time slot t in interval i
$c(n^v, i, t)$ and $b(e^v, i, t)$	the actual amounts of computing and bandwidth resource demands of n^v and e^v at time slot t in interval i
$\tau(G^v)$	the duration of virtual network G^v
$\mathbb{R}(G^v)$	the revenue of embedding virtual network G^v
$\mathbb{C}(G^v)$	the cost of embedding virtual network G^v
c_c and c_b	the prices for unit computing and bandwidth resources
$l_{e^s}^{e^v}$	an indicator value shows whether e^v is embedded to a path in G^s and e^s is a link in the path
$\Phi(n^s)$ and $\Phi(e^s)$	the embedding metrics that model the abilities of n^s and e^s in embedding a virtual node and link
a and b	constants for the embedding metrics $\Phi(n^s)$ and $\Phi(e^s)$ with $a > 1$ and $b > 1$
$d(e^s)$	the 'length' of a substrate link e^s which equals to $\frac{1}{\Phi(e^s)}$ if $\Phi(e^s) > 0$ and 0 otherwise
p	a path in the substrate network G^s
$d(p)$	the total 'length' of all substrate links in path p, i.e., $d(p) = \sum_{e^s \in p} d(e^s)$
$L(n^s)$	the set of substrate links incident to substrate node n^s
N_{sel}^s	candidate substrate nodes that are selected to embed virtual nodes in a virtual network
N_{emd}^v	virtual nodes that have been embedded into G^s
$NR(n^s)$ and $NR(n^v)$	the ranks of n^s and n^v that are used to select the cluster centers in G^s and G^v
n_c^s and n_c^v	the cluster centers in G^s and G^v, which have the highest values of $NR(n^s)$ and $NR(n^v)$
$\kappa(n^s)$ and $\kappa(n^v)$	the ranks of substrate nodes in G^s except n_c^s, and virtual nodes in G^v except n_c^v

of periodic resource demands of the virtual networks for a given monitoring period I, such that the revenue of the cloud service provider of G^s is maximized, subject to the constraint that the *resource demand violation ratio* of each virtual network G^v is bounded within its threshold $\sigma(G^v)$, where the resource demand violation ratio of a virtual network G^v is the amount of its violated resource demands to the total amount of its resource demands throughout its duration $\tau(G^v)$. For example, given a virtual network demanding one unit of resource at each time slot of its 10-time-slot lifetime, its resource demand violation ratio will be 10% if it is provided with 0.5 unit resource for two time slots and one unit for the rest.

Table 1 summarizes the symbols used in this paper.

4 Algorithm with the Knowledge of Periodic Resource Demands

In this section we consider virtual network embedding with the knowledge of periodic resource demands. We first devise an algorithm to embed a single virtual network $G^v = (N^v, E^v)$ to a substrate network $G^s = (N^s, E^s)$. We then propose an algorithm to embed multiple virtual networks to G^s. We finally analyze the time complexity of the proposed algorithms.

4.1 Embedding a virtual network with static and dynamic resource demands

To embed virtual network G^v to substrate network G^s, an embedding metric is needed. Such a metric captures not only the amounts of available resources but also the utilization ratios of the resources in G^s. In the following we first introduce a novel embedding metric. We then devise an embedding algorithm for embedding a virtual network with static resource demands, based on the proposed metric. We finally extend the algorithm to embed a virtual network with periodic resource demands.

We start by proposing an embedding metric to capture the workloads of substrate nodes and links in G^s. For a given substrate node n^s, the amount of available computing resource at it and its utilization ratio will jointly determine the *embedding ability* of substrate node n^s. *The marginal gain* of the embedding ability of n^s will diminish with the increase of its utilization ratio, since the larger the proportion of its computing resource is being occupied, the higher the risk of the SLA violations substrate node n^s faces. We thus use an exponential function to model the embedding ability of substrate node n^s. Recall that $P(n^s, i, t)$ is the amount of available computing resource of substrate node n^s at time slot t in interval i. Denote by $\Phi(n^s)$ the embedding metric of n^s, then

$$\Phi(n^s) = P(n^s, i, t) \cdot a^{\frac{P(n^s, i, t)}{C(n^s)}}, \tag{3}$$

where a is a constant with $a > 1$, and $\frac{P(n^s, i, t)}{C(n^s)}$ is the complementary ratio to the resource utilization ratio of n^s.

The defined embedding metric $\Phi(n^s)$ favors allocating a virtual node to a substrate node that has a large amount of available resource and a low resource utilization ratio. The embedding ability $\Phi(e^s)$ of a substrate link e^s can be defined similarly,

$$\Phi(e^s) = P(e^s, i, t) \cdot b^{\frac{P(e^s, i, t)}{B(e^s)}}, \tag{4}$$

where $b > 1$ is a constant and $P(e^s, i, t)$ is the amount of available bandwidth resource on e^s.

Having defined the embedding abilities of substrate links and nodes, we now introduce an embedding metric to embed each virtual link e^v to a substrate path p in G^s that consists of one or multiple substrate links. To this end, we first define the 'length' $d(e^s)$ of each substrate link e^s as follows.

$$d(e^s) = \begin{cases} \frac{1}{\Phi(e^s)} & \text{if } \Phi(e^s) > 0, \\ \infty & \text{if } \Phi(e^s) = 0. \end{cases} \quad (5)$$

The length of a substrate link implies that the shorter the substrate link, the more available bandwidth and lower utilization ratio the link will have. In other words, a substrate link without any available bandwidth will have a longest length, and should not be used by any routing path. Similarly, the 'length' $d(p)$ of a substrate path p is the sum of lengths of its constituent substrate links, which is defined as follows.

$$d(p) = \sum_{e^s \in p} d(e^s). \quad (6)$$

A shorter path p will have more available bandwidth on its constituent substrate links, thereby having a higher embedding ability.

We now embed a virtual network $G^v(N^v, E^s)$ with static resource demands to the substrate network $G^s(N^s, E^s)$, using the proposed embedding metrics as follows.

We embed each virtual node in N^v to a different substrate node N^s, followed by embedding each virtual link in E^v to a substrate path in G^s. To embed virtual nodes in N^v, we construct a cluster of substrate nodes for the virtual nodes through adding substrate nodes into the cluster one by one. Specifically, we first identify a *cluster center* with the greatest embedding ability in G^s to embed a virtual node with the maximum resource demand in G^v, and then find other substrate nodes one by one iteratively with a shorter path to the selected substrate nodes, by adopting a strategy similar to the one in [44]. Intuitively, each added substrate node in the cluster has not only the great embedding ability but also a shorter path to the other substrate nodes in the cluster, since a shorter path between two selected substrate nodes implies the more available bandwidth between them. Therefore, to find the cluster center, we assign each substrate node $n^s \in N^s$ *a rank* that is jointly determined by its embedding ability and the accumulative embedding ability of the links incident to it. Let $L(n^s)$ be the set of substrate links incident to substrate node $n^s \in N^s$. *The rank* $NR(n^s)$ of n^s is defined as *the product of* its embedding ability $\Phi(n^s)$ and the accumulative embedding ability of links in $L(n^s)$, i.e.,

$$NR(n^s) = \Phi(n^s) \cdot \sum_{e^s \in L(n^s)} \Phi(e^s). \quad (7)$$

The rank $NR(n^v)$ of each virtual node $n^v \in N^v$ can be defined similarly as the product of its computing resource demand and the accumulative bandwidth resource demand of the virtual links incident to it, i.e.,

$$NR(n^v) = C(n^v) \cdot \sum_{e^v \in L(n^v)} B(e^v). \quad (8)$$

Let n_c^s and n_c^v be the chosen cluster center and virtual node by Eqs. (7) and (8), respectively. Then, n_c^v is embedded into n_c^s. Having embedded n_c^v to n_c^s, the other $|N^v| - 1$ cluster members will be identified one by one iteratively. During each iteration, a virtual node is embedded to a substrate node with high embedding ability and the minimum distance to one of the selected substrate nodes in the cluster. To this end, rank those not yet selected substrate nodes by the product of the inverse of $\Phi(n^s)$ and the accumulative length from substrate node n^s to all the substrate nodes selected. Denote by $\kappa(n^s)$ the rank of n^s, then,

$$\kappa(n^s) = \frac{1}{\Phi(n^s)} \cdot \sum_{m^s \in N_{sel}^s} d(p_{n^s,m^s}), \quad (9)$$

where N_{sel}^s is the set of selected substrate nodes in the cluster, and p_{n^s,m^s} is the shortest path between substrate nodes n^s and $m^s \in N_{sel}^s$.

The rank of a yet-to-be embedded virtual node $n^v \in N^v$ can be defined similarly, i.e.,

$$\kappa(n^v) = \frac{1}{C(n^v)} \cdot \sum_{m^v \in N_{emd}^v} \sum_{e^v \in p_{n^v,m^v}} d(e^v), \quad (10)$$

where N_{emd}^v denotes the set of embedded virtual nodes, $d(e^v)$ is the length of virtual link defined by $\frac{1}{B(e^v)}$, and p_{m^v,n^v} is the shortest path between virtual nodes $m^v \in N_{emd}^v$ and n^v (w.r.t. the accumulative length of its virtual links). The rationale behind the definition of length metric $d(e^v)$ is that the length of virtual link e^v is inversely proportional to its demand B_v. Intuitively speaking, a shortest path between an embedded virtual node and the next virtual node to be embedded is a path with the maximum accumulated resource demands on the path. A virtual node n^v with the minimum value of $\kappa(n^v)$ will be chosen as the next virtual node to be embedded to the substrate node n^s with the lowest rank $\kappa(n^s)$. If there is such a virtual node in G^v that has never been embedded after considering all substrate nodes, then G^v will not be admitted.

The embedding of virtual links in E^v can be dealt similarly. Let p be a shortest path in G^s between two substrate nodes for virtual link e^v. If p does not have enough bandwidth to meet the bandwidth demand of e^v, the substrate link in p with the minimum available bandwidth is then removed, the next shortest path will be found until there is not such a path. The detailed procedure for a single virtual network embedding is detailed in procedure `EmbedOneVN-Static`.

Having shown how to embed a virtual network with static resource demands to the substrate network G^s, we here show how to embed a virtual network $G^v(N^v, E^v)$ with periodic resource demands to G^s as follows.

We construct $T + 1$ auxiliary graphs with each having

Procedure 1 `EmbedOneVN-Static()`

Input: $G^v(N^v, E^v)$, $G^s(N^s, E^s)$
Output: Embed G^v or reject it

/* Stage one: embed virtual nodes */
Find the cluster center n_c^s with the maximum rank in substrate network G^s by Eq. (7);
Find the virtual node n_c^v with the maximum rank in virtual network G^v by Eq. (8);
Embed virtual node n_c^v to the cluster center n_c^s;
$N_{emd}^v \leftarrow \{n_c^v\}$; /* the set of embedded virtual nodes*/
$N_{sel}^s \leftarrow \{n_c^s\}$; /* the set of selected substrate nodes*/
Embed virtual nodes in $N^v - N_{emd}^v$ to G^s one by one iteratively. Within an iteration, a virtual node with the minimum $\kappa(n^v)$ is embedded into a substrate node with the minimum $\kappa(n^s)$;
if $N^s \setminus N_{sel}^s = \emptyset$ and $N^v \setminus N_{emd}^v \neq \emptyset$ **then**
    Reject G^v; **exit**;
/* Stage two: embed virtual links */
for each virtual link $e^v \in E^v$ **do**
    Update the weight of substrate link e^s by Eq. (5);
    Let n_1^s and n_2^s be the substrate nodes that embed the two virtual nodes connected by e^v;
    Find a shortest path p from node n_1^s to n_2^s;
    if p cannot satisfy the resource demand $B(e^v)$ of e^v **then**
        Find the next shortest path from n_1^s to n_2^s;
    if no path can satisfy the demand of e^v **then**
        Reject G^v; **exit**;
Embed the virtual network G^v.

different resource demands at a different time slot in one interval, and 'pre-embed' each of the graphs by procedure `EmbedOneVN-Static`. Virtual network G^v will be embedded to G^s exactly by adopting one of the $T+1$ pre-embeddings that leads to the maximum revenue. Specifically, we construct T auxiliary graphs with each having the resource demands of G^v at time slot t' in an interval with $1 \leq t' \leq T$, and another auxiliary graph G_m^v having the average resource demands within an interval. Denote by $G^v(t') = (N_{t'}^v, E_{t'}^v)$ the auxiliary graph with resource demands of G^v at time slot t'. Then, $G^v(t')$ is constructed by setting $N_{t'}^v = N^v$, $E_{t'}^v = E^v$, $C(n^v) = c(n^v, i', t')$ for each $n^v \in N_{t'}^v$, $B(e^v) = b(e^v, i', t')$ for each $e^v \in E_{t'}^v$, where i' is one interval of the duration $\tau(G^v)$ of G^v. Similarly, $G_m^v = (N_m^v, E_m^v)$ with $N_m^v = N^v$ and $E_m^v = E^v$ denotes the virtual network G^v with average resource demands within an interval, i.e., the demands of its each virtual node and link are as follows.

$$C(n^v) = \frac{1}{T}\sum_{t'=1}^{T} c(n^v, i', t'), \text{ for each } n^v \in N_m^v, \quad (11)$$

and

$$B(e^v) = \frac{1}{T}\sum_{t'=1}^{T} b(e^v, i', t'), \text{ for each } e^v \in E_m^v. \quad (12)$$

Let $\mathcal{G}_{pre}$ be the set of constructed $T+1$ graphs, i.e., $\mathcal{G}_{pre} = \{G^v(t') \mid 1 \leq t' \leq T\} \cup \{G_m^v\}$. A pre-embedding of a graph in $\mathcal{G}_{pre}$ is *feasible* only if it can be embedded into G^s while the resource demands of G^v at each time slot are met. Let G_{max}^v be the graph among the $T+1$ auxiliary graphs that results in the maximum revenue. The detailed algorithm is given by procedure `EmbedOneVN-Periodic`.

Procedure 2 `EmbedOneVN-Periodic()`

Input: $G^v(N^v, E^v)$, $G^s(V^s, E^s)$
Output: Embed G^v or reject it

Construct $T+1$ auxiliary graphs for G^v with the first T graphs corresponding to its resource demands at each of the T time slots, and the last one denotes the average resource demand in one resource demand period. Denote by $\mathcal{G}_{pre}$ be the set of the $T+1$ graphs, i.e., $\mathcal{G}_{pre} \leftarrow \{G^v(t') \mid 1 \leq t' \leq T\} \cup \{G_m^v\}$;
$G_{max} \leftarrow NIL$; /*$G_{max} \in \mathcal{G}_{pre}$ is the graph that achieves the maximum revenue*/
$\mathbb{R}_{max} \leftarrow 0$ /*the revenue by the embedding of G_{max}*/;
for each graph G in $\mathcal{G}_{pre}$ **do**
    Pre-embed each graph in $\mathcal{G}_{pre}$ by invoking `EmbedOneVN-Static`, where a pre-embedding is feasible if the embedding satisfies the resource demands of G^v at each time slot in an interval;
    Calculate the revenue $\mathbb{R}(G)$ of the pre-embedding of G;
    if $\mathbb{R}_{max} < \mathbb{R}(G)$ **then**
        $\mathbb{R}_{max} \leftarrow \mathbb{R}(G)$;
        $G_{max} \leftarrow G$;
if all graphs in $\mathcal{G}_{pre}$ are rejected **then**
    Reject G^v;
    exit;
Embed G^v to G^s by the embedding of G_{max} with the maximum revenue.

4.2 Algorithm for embedding multiple virtual networks with identical resource demand periods

We now embed a set $\mathcal{G}(i,t)$ of virtual networks with identical resource demand periods to a substrate network G^s at time slot t within an interval i. To be specific, we first embed a virtual network $G^{v_1} \in \mathcal{G}(i,t)$ that results in the maximum revenue. Let $\mathcal{G}_2(i,t) = \mathcal{G}(i,t) - \{G^{v_1}\}$. We then embed the next virtual network in $\mathcal{G}_2(i,t)$ that leads to the maximum revenue, and so on. This procedure continues until either $\mathcal{G}_k(i,t)$ is empty or none of virtual networks in it can be embedded due to lack of cloud resources.

The detailed description of the embedding algorithm is given by `Algorithm 1`.

Algorithm 1 *Embedding a set $\mathcal{G}(i,t)$ of virtual networks with identical resource demand periods*

Input: G^s, $\mathcal{G}(i,t)$
Output: Virtual networks in $\mathcal{G}(i,t)$ to be embedded into G^s

All virtual networks in $\mathcal{G}(i,t)$ are unmarked;
while $\mathcal{G}(i,t) \neq \emptyset$ or there are unmarked virtual networks **do**
    $G_{max}^v \leftarrow NIL$; /*$G_{max}^v \in \mathcal{G}(i,t)$ is the virtual network that achieves the maximum revenue*/
    $\mathbb{R}_{max} \leftarrow 0$ /* the revenue by the embedding of G_{max}^v*/;
    for each virtual network $G^v \in \mathcal{G}(i,t)$ **do**
        Pre-embed G^v by invoking `EmbedOneVN-Periodic`;
        if G^v is rejected by `EmbedOneVN-Periodic` **then**
            G^v is marked as unembeddable;
        else
            Let $\mathbb{R}(G^v)$ be the revenue by embedding of G^v;
            if $\mathbb{R}_{max} < \mathbb{R}(G^v)$ **then**
                $\mathbb{R}_{max} \leftarrow \mathbb{R}(G^v)$;
                $G_{max}^v \leftarrow G^v$;
    Embed G_{max}^v by invoking `EmbedOneVN-Periodic`;
    $\mathcal{G}(i,t) \leftarrow \mathcal{G}(i,t) \setminus \{G_{max}^v\}$.

4.3 Algorithm for embedding multiple virtual networks with different resource demand periods

So far we have assumed that all virtual networks have identical resource demand periods. In reality, different virtual networks may have different resource demand

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.2016.2535215, IEEE Transactions on Cloud Computing

periods. For example, some enterprise IT services (e.g., virtual desktop services) have weekly resource demands, whereas others (such as pay-roll services) have fortnightly or monthly resource demands. We here deal with this general case of the problem by extending `Algorithm 1` to solve it.

Given a set $\mathcal{G}(i,t)$ of virtual networks with different resource demand periods at time slot t in interval i, the basic idea of the proposed algorithm is to classify virtual networks in $\mathcal{G}(i,t)$ into different categories (types), by their resource demand periods, and each virtual network in the same type x will have the same resource demand period T_x. Denote by X the number of different types of virtual networks. Intuitively, the more types of virtual networks, the more difficult embedding these virtual networks to the substrate network will be. To reduce the number of types (i.e., the number of resource demand periods), we merge one type of virtual networks with a shorter resource demand period to another type with a longer resource demand period if the longer one is divisible by the shorter one. The rationale behind this is that the longer resource demand period can be considered as a *super-period* that consists of multiple shorter periods. For example, a virtual network with weekly resource demands can be seen as a virtual network with fortnightly resource demands. Let T_{max} be the maximum resource demand period among the virtual networks in $\mathcal{G}(i,t)$. The merge procedure iteratively finds the maximum number of resource demand periods having a Least Common Multiple (LCM) T_{lcm} that is no greater than T_{max}, and merge the types corresponding to the found resource demand periods into one type that has resource demand period T_{lcm}. This merge procedure continues until no further merge is possible. Let Y be the number of types after the merging. We then proceed embedding virtual networks in $\mathcal{G}(i,t)$ type by type, starting with the type of virtual networks with the longest resource demand period. That is, for virtual networks in type T_y we embed them by invoking `Algorithm 1` for all y with $1 \leq y \leq Y$. The detailed algorithm description is given in `Algorithm 2`.

Algorithm 2 *Embedding a set $\mathcal{G}(i,t)$ of virtual networks with different resource demand periods to the cloud.*

Input: G^s, $\mathcal{G}(i,t)$, $T(G^v)$ for each $G^v \in \mathcal{G}(i,t)$
Output: Virtual networks in $\mathcal{G}(i,t)$ to be embedded into G^s

1: Classify virtual networks in $\mathcal{G}(i,t)$ into different types of classes by their resource demand periods;
2: For each resource demand period T_x, if there is a larger $T_{x'}$ that is a multiple of T_x, merge the type of virtual networks with the resource demand period T_x to the type one with the resource demand period $T_{x'}$;
3: Sort sequences of different types of virtual networks into increasing order by resource demand periods, i.e., $T_1, T_2, \ldots, T_{max}$;
4: **while** $LCM(T_1, T_2) > T_{max}$ **do**
5: Find the maximum number of resource demand periods having a Least Common Multiple (LCM) T_{lcm} that is no greater than T_{max};
6: Replace the found resource demand periods with T_{lcm};
7: Get the number Y of different types of resource demand periods;
8: **for** $y \leftarrow 1$ to Y **do**
9: Embed virtual networks with the resource demand period T_y by invoking `Algorithm 1`;

4.4 Algorithm analysis

The rest is to analyze the time complexity of the proposed embedding algorithms `Algorithm 1` and `Algorithm 2` for multiple virtual network embedding.

Theorem 1: Given a distributed cloud $G^s = (V^s, E^s)$ and a monitoring period consisting of I intervals with each having T equal time slots, let $\mathcal{G}(i,t)$ be the set of virtual network requests arrived in the beginning of time slot t in interval i with each having an identical resource demand period. There is an algorithm, `Algorithm 1`, for the virtual network embedding problem with the knowledge of identical resource demand periods, which takes $O(\sum_{i=1}^{I} |\mathcal{G}(i,t)|^2(|N^v_{max}||E^s||N^s|^2 + |N^s|^3))$, where $|\mathcal{G}(i,t)|$ is the number of virtual network requests in $\mathcal{G}(i,t)$, $|N^v_{max}|$ is the maximum number of virtual nodes of a virtual network for each i and t, with $1 \leq i \leq I$ and $1 \leq t \leq T$.

Proof: We start by analyzing the time complexity of procedure `EmbedOneVN-Static` that consists of two stages. Recall that, in stage one, virtual nodes in each virtual network $G^v(N^v, E^v) \in \mathcal{G}(i,t)$ are embedded into substrate nodes in the substrate network $G^s(N^s, E^s)$ by iteratively selecting a substrate node with the highest rank $\kappa(n^s)$ for the virtual node n^v with the highest rank $\kappa(n^v)$. To calculate $\kappa(n^s)$ and $\kappa(n^v)$ for each substrate node n^s and virtual node n^v, all pairs of shortest paths in both G^s and G^v are found, which takes $O(|N^s|^3)$ and $O(|N^v|^3)$ time, respectively. In addition, finding a substrate node for each virtual node in N^v takes $O(|N^v||N^s|)$ time. Stage one thus takes $O(|N^s|^3 + |N^v|^3 + |N^v||N^s|) = O(|N^s|^3)$ time, since $|N^v| << |N^s|$. In stage two, each virtual link e^v is embedded into a substrate path in G^s. Since the shortest path between two substrate nodes that correspond to the two endpoints of e^v may not have enough bandwidth to meet the bandwidth requirement of e^v, the bottleneck edge with the minimum available bandwidth in the path then is removed, and the next shortest path for each virtual link e^v is then found. This procedure continues until a shortest path with enough available bandwidth is found, or no such a shortest path exists. If there is no such a path meeting the bandwidth requirement, the virtual network request will be rejected. This procedure takes $O(|E^s||N^s|^2)$ time, as there are $|E^v|$ virtual links in G^v, stage two takes $O(|E^v||E^s||N^s|^2)$ time. Thus, procedure `EmbedOneVN-Static` takes $O(|N^v||E^s||N^s|^2 + |N^s|^3)$ time.

We then analyze the time complexity of procedure `EmbedOneVN-Periodic`, this procedure treats each virtual network $G^v(N^v, E^v)$ with periodic resource demands as $T+1$ virtual networks with static resource demands at each time slot as well as the average resource demand at each interval. These $T+1$ virtual networks are 'pre-embedded', and G^v will be embedded exactly as one of these $T+1$ pre-embeddings that leads to the maximum

revenue. Thus, embedding a virtual network G^v with periodic resource demands takes $O(T(|N^v||E^s||N^s|^2 + |N^s|^3))$ time.

We finally analyze the time complexity of `Algorithm 1`. There are $|\mathcal{G}(i,t)|$ virtual networks with periodic resource demands. Recall that `Algorithm 1` chooses a virtual network that will lead to the maximum revenue among yet-to-be embedded virtual networks in $\mathcal{G}(i,t)$. To find the virtual network with the maximum revenue, the algorithm pre-embeds each virtual network by invoking procedure `EmbedOneVN-Periodic`. In total, there are $O(|\mathcal{G}(i,t)|^2)$ attempts of calling the procedure. Thus, `Algorithm 1` takes $O(|\mathcal{G}(i,t)|^2 T(|N^v||E^s||N^s|^2 + |N^s|^3))$ time to embed all virtual networks in $\mathcal{G}(i,t)$, which is $O(|\mathcal{G}(i,t)|^2|(|N^v_{max}||E^s||N^s|^2 + |N^s|^3))$ as only the maximum number $|N^v_{max}|$ of virtual nodes of a virtual network in $\cup_{i=1}^{I} \cup_{t=1}^{T} \mathcal{G}(i,t)$ needs to be considered and, T usually is constant, i.e., $T = O(1)$. The theorem holds. □

Theorem 2: Given a distributed cloud $G^s = (V^s, E^s)$ and a monitoring period consisting of I intervals with each having T equal time slots, let $\mathcal{G}(i,t)$ be the set of arrived virtual networks in the beginning of time slot t in interval i with each having a different resource demand period $T(G^v)$. There is an algorithm, `Algorithm 2`, for the virtual network embedding problem with the knowledge of different resource demand periods, which takes $O(\sum_{i=1}^{I} |\mathcal{G}(i,t)|^3(|N^v_{max}||E^s||N^s|^2 + |N^s|^3))$ time, where $|\mathcal{G}(i,t)|$ is the number of virtual network requests in $\mathcal{G}(i,t)$, $|N^v_{max}|$ is the maximum number of virtual nodes of a virtual network for each t and i with $1 \leq i \leq I$ and $1 \leq t \leq T$.

Proof: The proof of Theorem 2 is similar to the proof of Theorem 1, omitted. □

5 Algorithm without the Knowledge of Periodic Resource Demands of Virtual Network Requests

The proposed algorithms so far assumed that the periodic resource demands of each virtual network are given in advance. In reality, very few users know the resource demand periods of their virtual networks. Instead, users normally just specify their maximum resource demands. If the cloud service provider embeds each virtual network by user specified maximum resource demands, the resource utilization will be low, as shown in Fig. 1. Alternatively, the cloud service provider can allocate its resources intelligently by predicting the periodic resource demands of each admitted virtual network. This can be achieved through analyzing the historic resource demands of the virtual network. In the following we propose a prediction algorithm for this purpose.

The basic idea behind the prediction algorithm is to embed a newly admitted virtual network to meet its maximum resource demands initially, by procedure `EmbedOneVN-Static`. The algorithm adjusts the amounts of the resources allocated to the virtual network periodically. Specifically, let $K(G^v)$ be the number of intervals after which the resource demands of a virtual network G^v will be adjusted. The amounts of resources allocated to G^v thus will be adjusted every $K(G^v)$ intervals until its duration $\tau(G^v)$ expires. In total, there will be $\lfloor \frac{\tau(G^v)}{K(G^v)} \rfloor$ adjustments of demanded resources for an embedded virtual network G^v during its lifetime $\tau(G^v)$.

The key in the prediction is how to adjust the amounts of resources allocated to G^v. To this end, we record its actual resource demands of G^v at the past $K(G^v)$ intervals, and use these historic data to predict its resource demands in the current interval i. We then allocate the predicted amounts of resources to G^v. Recall that $\hat{c}(n^v,i,t)$ is the predicted computing resource demand of virtual node n^v at time slot t in interval i, which can be derived by an autoregressive moving average prediction method [25] as follows.

$$\hat{c}(n^v,i,t) = \sum_{k=1}^{K(G^v)} \beta_{i-k} c(n^v, i-k, t), \tag{13}$$

where β_{i-k} is a given constant related to the resource demands in interval $i-k$, and $\sum_{k=1}^{K(G^v)} \beta_{i-k} = 1$ with $\beta_{i-k} \geq \beta_{i-k-1}$ and $0 < \beta_{i-k} < 1$. This prediction model gives an insight that the resource demands G^v at the current interval i are related to its resource demands in its previous $K(G^v)$ intervals. The embedding algorithm is described by `Algorithm 3`.

Algorithm 3 *Embedding a set $\mathcal{G}(i,t)$ of virtual networks with identical resource demand periods and without the periodic resource demands.*

Input: G^s, $\mathcal{G}(i,t)$, $K(G^v)$ for each $G^v \in \mathcal{G}(i,t)$
Output: Virtual networks in $\mathcal{G}(i,t)$ to be embedded into G^s and resource allocation adjustments for embedded virtual networks

1: Embed each $G^v \in \mathcal{G}(i,t)$ by allocating it with its maximum resource demands by invoking `EmbedOneVN-Static`;
2: **for** each embedded virtual network G^v_{emd} in G^s **do**
3: Calculate the number of intervals that G^v_{emd} spanned in G^s so far, and let i' be the number;
4: **if** $i' > 0$ and $(i' \mod K(G^v)) = 0$ **then**
5: Predict the computing resource demand of each n^v of its virtual nodes, $\hat{c}(n^v,i,t')$, and the bandwidth resource demand of each e^v of its virtual links, $\hat{b}(e^v,i,t')$ by Eq. (13), for the current interval i, where $1 \leq t' \leq T$;
6: Reserve $\hat{c}(n^v,i,t)$, the amount of computing resource for n^v, and $\hat{b}(e^v,i,t)$, the amount of bandwidth resource for e^v in the next $K(G^v)$ intervals;

Notice that it is sufficient to modify step 9 of `Algorithm 2` by invoking `Algorithm 3` in order to embed a virtual network G^v with different resource demand periods and without the knowledge of periodic resource demands. Due to space limitation, the description of this algorithm is omitted.

6 Experimental Study

In this section, we evaluate the performance of the proposed algorithms and investigate the impact of different parameters on the algorithm performance.

6.1 Simulation settings

We adopt both synthetic and real network topologies for the substrate network G^s. Specifically, we generate G^s by the GT-ITM tool [19], which consists of 50 substrate nodes, and there is an edge between each pair of nodes with a probability of 0.1, following the similar settings in [10], [39], [44]. We also adopt a real network topology, GÉANT consisting of 40 nodes and 61 edges for the substrate network G^s [17]. The computing capacity of each substrate node in G^s is randomly drawn from 2,000 GHz to 5,000 GHz, and the bandwidth capacity of each substrate link in G^s is drawn from 10 Mbps to 1,000 Mbps [2], [11], [20], [27]. The number of virtual nodes of each virtual network varies from 2 to 10, and there is a virtual link between every two virtual nodes with a probability of 0.5. Parameters a and b in Eq. (3) are set to 5, which will be explained later. The number of time slots of each interval is $T = 7$, e.g., 7 days a week. The monitoring period consists of 100 time slots. The arrival rate of virtual network requests follows the Poisson process with an average rate of 5 virtual networks per time slot, and the duration of each virtual network varies with no more than 50 intervals. The length of $K(G^v)$ in `Algorithm 3` is set to 5.

Resource demands of virtual networks: For resource demands of virtual networks within an interval, we consider two patterns: one is the `Weekday-weekend` pattern, where a virtual network has demand peaks during either weekdays (5 demand peaks) or weekends (2 demand peaks); another is the `Random` pattern, where the number of time slots when a virtual network has peak demands is randomly generated. In both resource demand patterns, the amounts of peak computing and bandwidth demands are randomly generated from 2 to 10. The off-peak resource demands are no more than 80% of its peak resource demands.

Benchmarks: We evaluate the proposed algorithms for the VNE problem against two state-of-the-art algorithms. The first one is the algorithm in [44], referred to as algorithm `MAX`, in which the embedding ability of each substrate node n^s is the amount of its available computing resource, i.e., $\Phi(n^s) = P(n^s, i, t)$, it embeds virtual networks to G^s according to their maximum resource demands. The second one, referred to as `PAGERANK`, is a PageRank-based embedding algorithm [8], which assigns each virtual node and each substrate node a rank by adopting the PageRank algorithm. It then embeds each virtual network by mapping its virtual nodes, followed by embedding its virtual links to the substrate network. For simplicity, we use algorithms `ALG-PERIOD`, and `ALG-NO-PERIOD` to denote `Algorithm 1` and `Algorithm 3` with and without the knowledge of periodic resource demands, respectively. Similarly, we use `ALG-PERIOD-DIFF` and `ALG-NO-PERIOD-DIFF` to denote `Algorithm 2` and its version without the knowledge of periodic resource demands.

Evaluation metrics: In addition to the revenue achieved by embedding of virtual networks, we also consider the revenue-to-cost ratio $\eta(G^v)$ to quantify the efficiency of embedding of each virtual network G^v as follows.

$$\eta(G^v) = \frac{\mathbb{R}(G^v)}{\mathbb{C}(G^v)}. \tag{14}$$

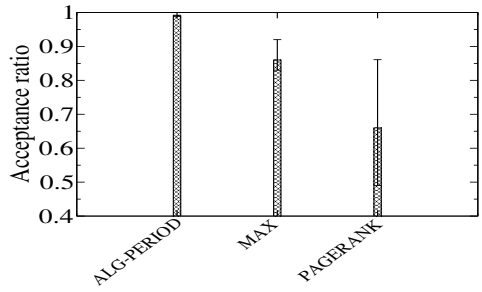
Given a distributed cloud $G^s = (V^s, E^s)$ and a monitoring period I, the *accumulated revenue, service cost, and revenue-to-cost ratio* within the monitoring period thus are the total revenues collected, the total service cost spent, and the ratio of the total revenue to the total service cost for the monitoring period. Each value in our figures is the mean of the results by applying the mentioned algorithm to either 15 synthetic network topologies or 15 resource capacity settings of the GÉANT topology. Also, 95% confidence intervals for these mean values are presented in all figures.

6.2 Performance evaluation

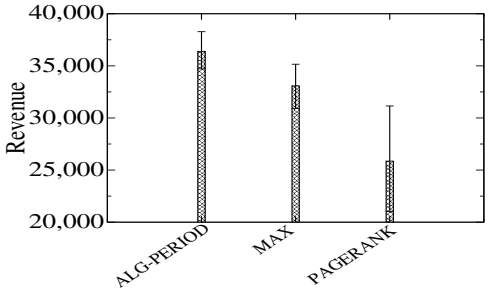
We first evaluate two proposed algorithms `ALG-PERIOD` and `ALG-NO-PERIOD` against algorithms `MAX` and `PAGERANK`, based on the synthetic substrate networks generated by GT-ITM. Fig. 3 shows the results when virtual networks follow the `Random` resource demand pattern. Specifically, Fig. 3(a) indicates that, on average, algorithm `ALG-PERIOD` admits around 15% and 30% more requests than those of algorithms `MAX` and `PAGERANK` in a monitoring period consisting of 100 time slots. It can also be seen from Fig. 3(b) that algorithm `ALG-PERIOD` earns 10% and 31% more revenues than these of algorithms `MAX` and `PAGERANK`. The reason behind is that algorithm `ALG-PERIOD` performs fine-grained resource allocations by exploring the periodic resource demands and adopting a novel embedding metric that takes into account both the dynamic workload on the substrate network and the user periodic resource demands.

It can be seen from Figures 3(a) and (b) that algorithms `MAX` and `PAGERANK` have wider confidence intervals for the mean of acceptance ratios and revenues. The reason is that they allocate cloud resources according to the maximum demands of virtual networks. Their acceptance ratios or revenues will oscillate a lot since each virtual network request is likely to be rejected with high probability, and algorithm `PAGERANK` is the worst among them. Figures 3(c) and 3(d) clearly demonstrate that algorithm `ALG-NO-PERIOD` consistently delivers the highest acceptance ratio and the maximum number of admitted virtual network requests among the three mentioned algorithms. It can be seen from Fig. 3(c) that the virtual network acceptance ratios of algorithms `MAX` and `PAGERANK` oscillate a lot, since the number of admitted virtual networks is only affected by the amount of available resources in the substrate network. In addition, Fig. 3(e) implies that the revenue delivered by algorithm `ALG-NO-PERIOD` is higher than that by algorithms `MAX` and `PAGERANK`, and it has also the higher revenue-to-

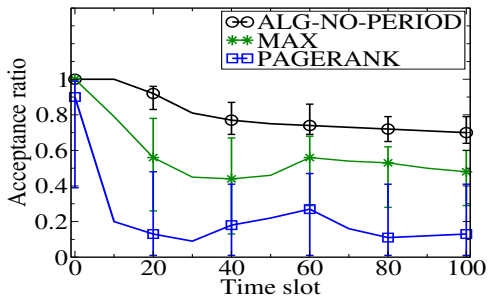
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2535215, IEEE Transactions on Cloud Computing



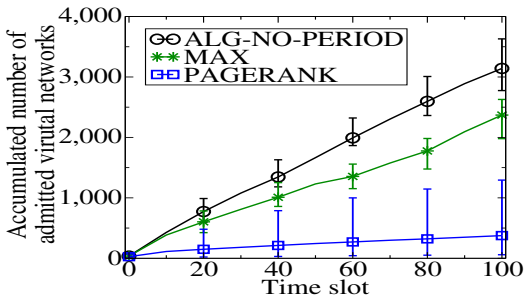
(a) The acceptance ratio



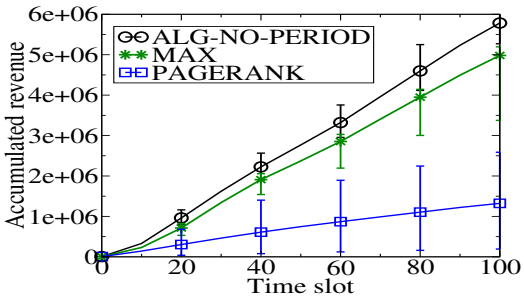
(b) The revenue



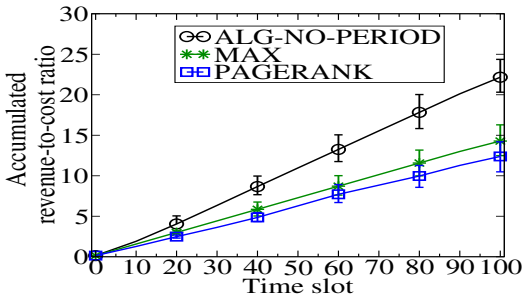
(c) The acceptance ratio



(d) The accumulated number of admitted virtual networks



(e) The accumulated revenue



(f) The accumulated revenue-to-cost ratio

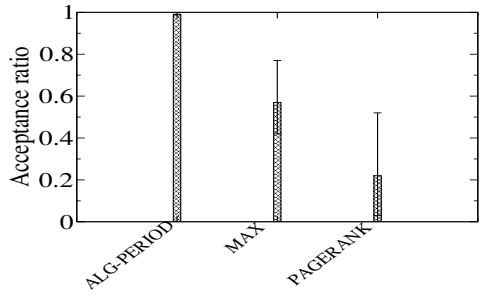
Fig. 3: The performance of different algorithms under `Random` resource demand patterns of virtual networks

cost ratio than these of algorithms `MAX` and `PAGERANK` as shown in Fig. 3(f).

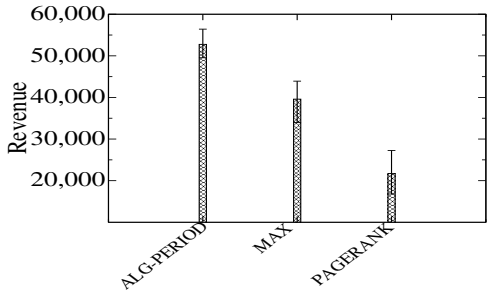
Fig. 4 depicts the performance curves when the resource demands of virtual networks follow the `Weekday-weekend` resource demand pattern, from which it can be seen that algorithms `ALG-PERIOD` and `ALG-NO-PERIOD` outperform the others in terms of the acceptance ratios, revenues, and revenue-to-cost ratios. In addition, the acceptance ratio of algorithm `ALG-PERIOD` with pattern `Weekday-weekend` (Fig. 4 (a)) is higher than that of it with pattern `Random` (Fig. 3 (a)), so is its revenue that can be seen from Figures 3(b) and 4(b), respectively. This is because that there are more opportunities for virtual networks with complimentary resource demands to share resources with each other. For the sake of clarity, in the rest of evaluation, we will focus only on the `Random` resource demand pattern.

We now evaluate algorithms `ALG-PERIOD-DIFF` and `ALG-NO-PERIOD-DIFF` against algorithms `MAX` and `PAGERANK`, by assigning the resource demand period of each virtual network with one of the values in $\{7, 14, 15, 30\}$. It can be seen from Fig. 5 that algorithms `ALG-PERIOD-DIFF` and `ALG-NO-PERIOD-DIFF` outperform algorithms `MAX` and `PAGERANK`. Furthermore, by comparing the results in Fig. 3 and Fig. 5, it can be seen that algorithms `ALG-PERIOD-DIFF` and `ALG-NO-PERIOD-DIFF` are inferior to their counterparts, `ALG-PERIOD` and `ALG-NO-PERIOD`. For example, as shown in Fig. 5(a), the acceptance ratio by algorithm `ALG-PERIOD-DIFF` is lower than that by algorithm `ALG-PERIOD` in Fig. 3(a), since algorithm `ALG-PERIOD-DIFF` merges the type of virtual networks with a shorter resource demand period to the type with a longer resource demand period that is a multiple of the shorter one. This means that the resource availability check process (Step 5 in procedure `EmbedOneVN-Periodic`) will consider a longer interval, thereby increasing the rejection probability of virtual networks. Furthermore, it can be seen from Figures 5(b) and 5(c) that the accumulated revenue of algorithm `ALG-NO-PERIOD-DIFF` is roughly the same as that of algorithm `ALG-NO-PERIOD`, whereas the accumulated revenue-to-cost ratio of algo-

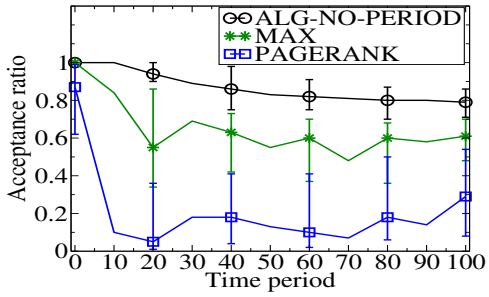
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2016.2535215, IEEE Transactions on Cloud Computing



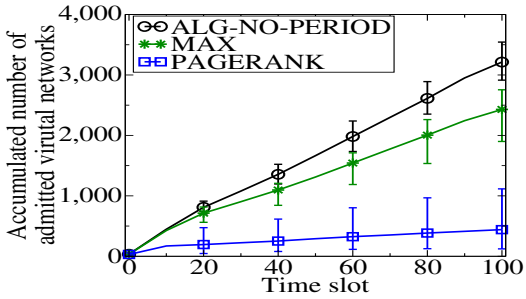
(a) The acceptance ratio



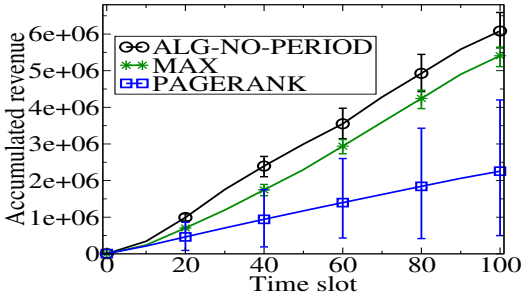
(b) The revenue



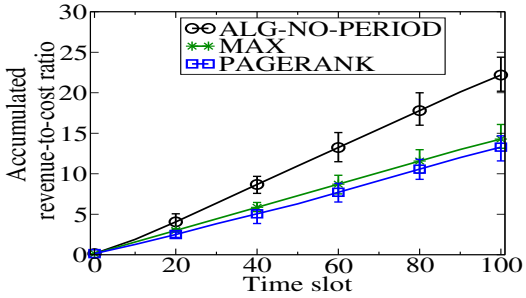
(c) The acceptance ratio



(d) The accumulated number of admitted virtual networks



(e) The accumulated revenue



(f) The accumulated revenue-to-cost ratio

Fig. 4: The performance of different algorithms under `Weekday-weekend` resource demand patterns

rithm `ALG-NO-PERIOD-DIFF` is much lower than that of algorithm `ALG-NO-PERIOD`. In other words, the cost by algorithm `ALG-NO-PERIOD-DIFF` is much higher than that by algorithm `ALG-NO-PERIOD`, because more substrate links are needed in order to embed each virtual link, given that the rejection probability of a virtual link increases.

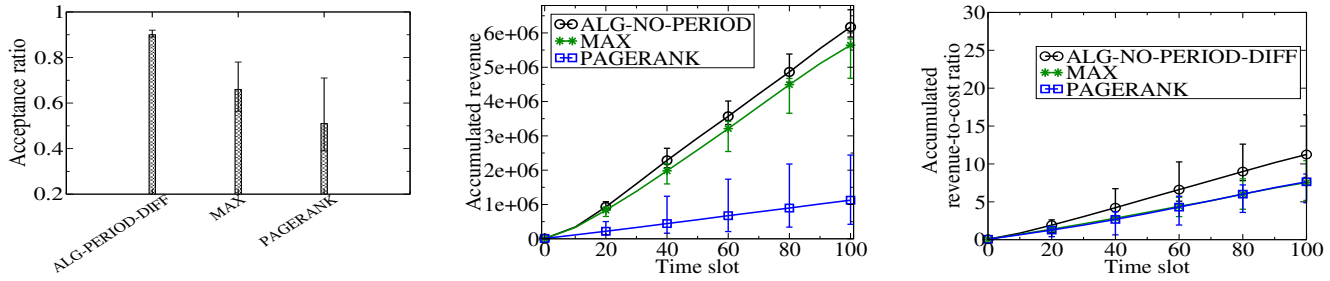
6.3 Performance evaluation using GÉANT topology

To evaluate the performance of the proposed algorithms in real networks, we now evaluate algorithms `ALG-PERIOD` and `ALG-NO-PERIOD` against algorithms `MAX` and `PAGERANK` in the GÉANT network. It can be seen from Figures 6(a) and 6(b) that algorithms `ALG-PERIOD` and `ALG-NO-PERIOD` outperform algorithms `MAX` and `PAGERANK` in terms of the acceptance ratio and revenue. Specifically, Fig. 6(b) shows that algorithm consistently achieves higher revenues than those of algorithms `MAX` and `PAGERANK`, and the overlapping on that by the confidence intervals by these algorithms are trivial. For example, the revenue by `ALG-NO-PERIOD` is around twice that of algorithm `PAGERANK`, and 30% more than that of algorithm `MAX`. Also, it must be mentioned that the acceptance ratios by all algorithms in GÉANT are lower than that in random topologies generated by GT-ITM. This is because the network size of GÉANT is smaller and has less number of edges compared with the topologies generated by GT-ITM. Also, it can be seen from Fig. 6(c) that algorithm `ALG-NO-PERIOD` outperforms algorithms `MAX` and `PAGERANK`. For example, within a monitoring period of 100 intervals, the accumulated revenue-to-cost ratio by algorithm `ALG-NO-PERIOD` is around 30% and 35% higher than those by algorithms `MAX` and `PAGERANK`, respectively.

6.4 Impact of parameters

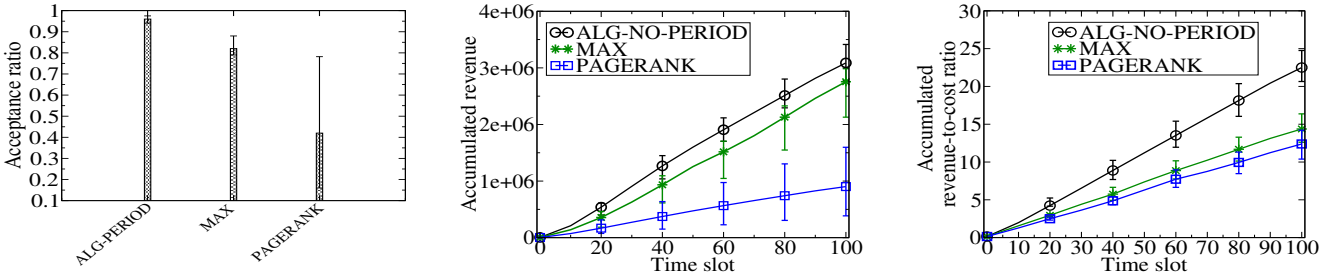
We then study the impact of constant parameters a and b in embedding metrics in Eqs. (3) and (4) on the performance of the proposed algorithms `ALG-NO-PERIOD` and `ALG-PERIOD`, by varying their values from 5 to 5^7. For the sake of simplicity, we only evaluate the algorithms for virtual networks with `Random` resource demand patterns. Figures 7(a) and 7(b) demonstrate that

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.2016.2535215, IEEE Transactions on Cloud Computing



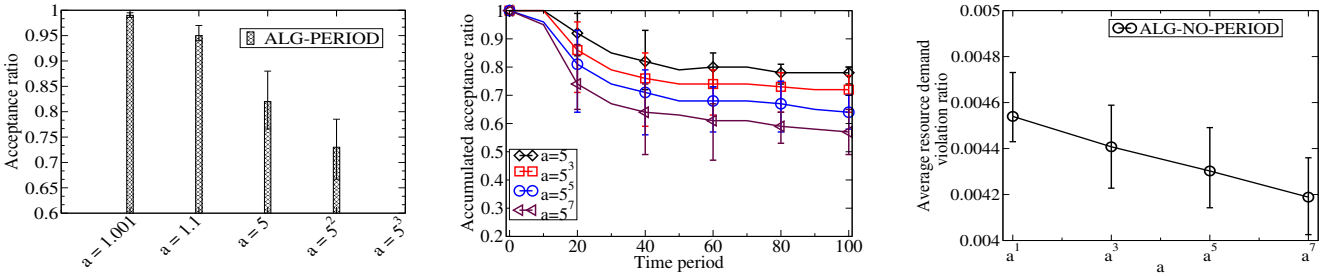
(a) The acceptance ratio (b) The accumulated revenue (c) The accumulated revenue-to-cost ratio

Fig. 5: The performance of different algorithms with different resource demand periods and under `Random` resource demand patterns of virtual networks



(a) The acceptance ratio (b) The accumulated revenue (c) The accumulated revenue-to-cost ratio

Fig. 6: The performance of different algorithms using GÉANT topology under `Random` resource demand patterns of virtual networks



(a) The impact of a on the acceptance ratio of algorithm `ALG-PERIOD` (b) The impact of a on the acceptance ratio of algorithm `ALG-NO-PERIOD` (c) The impact of a on the average resource demand violation ratio of algorithm `ALG-NO-PERIOD`

Fig. 7: The impact of a on the performance of different algorithms

the larger the value of a, the lower the acceptance ratio. This implies that when there is a larger a, each substrate node is reluctant to accommodate a virtual node when its utilization rate is nearly full, otherwise leading to SLA violations. Fig. 7(c) plots the resource demand violations by algorithm `ALG-NO-PERIOD` for different values of a. It can be seen that algorithm `ALG-NO-PERIOD` has the highest and lowest resource demand violation ratios when $a = 5$ and $a = 5^7$, respectively. The algorithm delivers a resource demand violation ratio less than 0.5% when $a = 5$, but achieves a relatively high acceptance ratio. This is why we set $a = 5$ in the default setting.

7 Conclusion

In this paper we considered virtual network embedding problems with and without the knowledge of periodic resource demands in a substrate network. We devised efficient embedding algorithms, by incorporating novel embedding metrics and periodic resource demands of virtual network requests, provided that periodic resource demands of each virtual network are given and all virtual networks have identical resource demand periods; otherwise, we proposed a period prediction method to predict the periodic resource demands of each admitted virtual network. We finally evaluated the performance of the proposed algorithms through experimental simulations, based on both synthetic and real substrate networks. Experimental results demonstrate that the proposed algorithms are promising, and outperform existing heuristics.

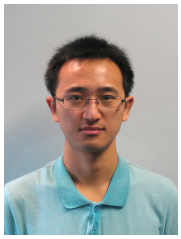
Acknowledgement

We appreciate the three anonymous referees and the Associate Editor for their constructive comments and valuable suggestions, which help us significantly improve the quality and presentation of the paper.

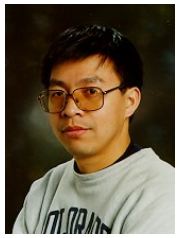
References

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner. Overcoming the Internet impasse through virtualization. *Computer*, Vol. 38, pp.34–41, IEEE, 2005.

[2] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards predictable datacenter networks. *Proc. ACM SIGCOMM*, 2011.
[3] M. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and M. Zhani. Data center network virtualization: a survey. *Communications Surveys &Tutorials*, Vol. 15, pp. 909–928, IEEE, 2013.
[4] T. Benson, A. Akella, A. Shaikh, and S. Sahu. CloudNaaS: a cloud networking platform for enterprise applications. *Proc. ACM SOCC*, 2011.
[5] R. Bradford, E. Kotsovinos, A, Feldmann, and H. Schiöberg. Live wide-area migration of virtual machines including local persistent state. *Proc. ACM VEE*, 2007.
[6] N. F. Butt, M. Chowdhury, and R. Boutaba. Topology-awareness and reoptimization mechanism for virtual network embedding. *Proc. IFIP NETWORKING*, Springer, 2010.
[7] Z. Cai, F. Liu, N. Xiao, Q. Liu, and Z. Wang. Virtual network embedding for evolving networks. *Proc. IEEE GLOBECOM*, 2010.
[8] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang. Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Computer Communication Review*, Vol. 41, pp. 38–47, 2011.
[9] N. M. M. K. Chowdhury and R. Boutaba. A survey of network virtualization. *Computer Networks*, Vol. 54, pp.862–876, Elsevier, 2010.
[10] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. *Proc. IEEE INFOCOM*, 2009.
[11] R. Duan, R. Prodan, and X. Li. Multi-objective game theoretic scheduling of bag-of-tasks workflows on hybrid clouds. *IEEE Trans. on Cloud Computing*, Vol. 2, pp. 29–42, 2014.
[12] F. Esposito, D. Paola, and I. Matta. On distributed virtual network embedding with guarantees. to appear in *IEEE/ACM Transactions on Networking*, IEEE, 2014.
[13] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann. Vnr algorithm: a greedy approach for virtual networks reconfiguration. *Proc. IEEE GLOBECOM*, 2011.
[14] N. Feamster, L. Gao, and J. Rexford. How to lease the Internet in your spare time. *ACM SIGCOMM Computer Communication Review*, Vol. 37, pp.61–64, ACM, 2007.
[15] P. Ferguson and G. Huston. What is a VPN? *Technique Report*, Cisco Systems, 1998.
[16] A. Fischer, J. Botero, M. Beck, H. Meer, and X. Hesselbach. Virtual network embedding: a survey. *Communications Surveys & Tutorials*, Preprint, IEEE, 2013.
[17] GÉANT. `http://www.geant.net`.
[18] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Workload analysis and demand prediction of enterprise data center applications. *Proc. IEEE IISWC*, 2007.
[19] GT-ITM. `http://www.cc.gatech.edu/projects/gtitm/`.
[20] C. Guo *et. al.* SecondNet: a data center network virtualization architecture with bandwidth guarantees. *Proc. ACM CONEXT*, 2010.
[21] J. Guo, F. Liu, D. Zeng, J. Lui, and H. Jin. A cooperative game based allocation for sharing data center networks. *Proc. IEEE INFOCOM*, 2013.
[22] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache. Virtual network provisioning across multiple substrate networks. *Computer Networks*, Vol. 55, pp.1011–1023, Elsevier, 2011.
[23] I. Houidi, W. Louati, and D. Zeghlache. A distributed and autonomic virtual network mapping framework. *Proc. IEEE ICAS*, 2008.
[24] I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, and L. Mathy. Adaptive virtual network provisioning. *Proc. ACM VISA*, 2010.
[25] A. Kansal, F. Zhao, N. Kothari, and A. A. Bhattacharya. Virtual machine power metering and provisioning. *Proc. ACM SoCC*, 2010.
[26] J. Lischka and H. Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. *Proc. ACM VISA*, 2009.
[27] C. Mastroianni, M. Meo, and G. Papuzzo. Probabilistic consolidation of virtual machines in self-organizing cloud data centers. *IEEE Trans. on Cloud Computing*, Vol. 1, pp. 215–228, 2013.
[28] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem. Adaptive control of virtualized resources in utility computing environments. *Proc. ACM EUROSYS*, 2007.
[29] A. Shieh, S. Kandulaz, A. Greenberg, C. Kim, and B. Saha. Sharing the data center network. *Proc. USENIX NSDI*, 2011.
[30] S. Su, Z. Zhang, A. X. Liu, X. Cheng, Y. Wang, and X. Zhao. Energy-aware virtual network embedding. IEEE/ACM Transactions on Networking, Vol. 22, pp.1607 – 1620, 2014.
[31] G. Sun, V. Anand, H. Yu, D. Liao, Y. Cai, and L. M. Li. Adaptive provisioning for evolving virtual network request in cloud-based data centers. *Proc. IEEE GLOBECOM*, 2012.
[32] U.S. Department of Energy. Creating energy efficient data centers. 2007.
[33] T. Wood, K. K. Ramakrishnan, P. Shenoy, and J. Merwe. Enterprise-Ready Virtual Cloud Pools: Vision, Opportunities and Challenges. *Oxford computer journal*, Vol. 55, pp.995–1004, 2012.
[34] D. Xie, N. Ding, Y. C. Hu, and R. Kompella. The only constant is change : incorporating time-varying network reservations in data centers. *Proc. ACM SIGCOMM*, 2012.
[35] Z. Xu and W. Liang. Operation cost minimization for distributed data centers through exploring electricity price diversity. *Computer Networks*, Vol. 83, pp. 59–75, Elsevier, 2015.
[36] Z. Xu and W. Liang. Minimizing the operational cost of data centers via geographical electricity price diversity. *Proc. IEEE CLOUD*, 2013.
[37] Z. Xu, W. Liang, and Q. Xia. Electricity cost minimization in distributed clouds by exploring heterogeneities of cloud resources and user demands. To appear in *Proc. IEEE ICPADS*, 2015.
[38] Z. Xu, W. Liang, and Q. Xia. Efficient virtual network embedding via exploring periodic resource demands. *Proc. IEEE LCN*, 2014.
[39] M. Yu, Y. Yi, J. Rexford, and M. Chiang. Rethinking virtual network embedding: substrate support for path splitting and migration. *ACM SIGCOMM Computer Communication Review*, Vol. 38, pp. 17–29, 2008.
[40] S. Zhang, Z. Qian, J. Wu, and S. Lu. An opportunistic resource sharing and topology-aware mapping framework for virtual networks. *Proc. IEEE INFOCOM*, 2012.
[41] S. Zhang and X. Qiu. A novel virtual network mapping algorithm for cost minimizing. *Journal of Selected Areas in Telecommunications*, Vol. 02, No. 01, pp. 1–9, 2011.
[42] S. Zhang, J. Wu, and S. Lu. Virtual network embedding with substrate support for parallelization. *Proc. IEEE GLOBECOM*, 2012.
[43] Y. Zhou, X. Yang, Y. Li, D. Jin, L. Su, and L. Zeng. Incremental re-embedding scheme for evolving virtual network requests. *IEEE Communication Letters*, Vol. 17, pp. 1016–1019, 2013.
[44] Y. Zhu and M. Ammar. Algorithms for assigning substrate network resources to virtual network components. *Proc. IEEE INFOCOM*, 2006.



Zichuan Xu received his ME degree and BSc degree from Dalian University of Technology in China in 2011 and 2008, both in Computer Science. He is currently pursuing his PhD study in the Research School of Computer Science at the Australian National University. His research interests include distributed clouds, data center networks, cloud computing, algorithmic game theory, and optimization problems.



Weifa Liang (M'99–SM'01) received the PhD degree from the Australian National University in 1998, the ME degree from the University of Science and Technology of China in 1989, and the BSc degree from Wuhan University, China in 1984, all in computer science. He is currently an Associate Professor in the Research School of Computer Science at the Australian National University. His research interests include design and analysis of routing protocols for wireless ad hoc and sensor networks, cloud computing, graph databases, design and analysis of parallel and distributed algorithms, combinatorial optimization, and graph theory. He is a senior member of the IEEE.

Qiufen Xia received her ME degree and BSc degree from Dalian University of Technology in China in 2012 and 2009, both in Computer Science. She is currently pursuing her PhD study in the Research School of Computer Science at the Australian National University. Her research interests include mobile cloud computing, distributed clouds, cloud computing, and optimization problems.