Computer Communications 121 (2018) 1–18

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Joint load balancing and energy saving algorithm for virtual network embedding in infrastructure providers

Chan Kyu Pyoung, Seung Jun Baek*

Department of Computer Science and Engineering, Korea University, Seoul 02841, Republic of Korea

ARTICLE INFO

Keywords:
Network virtualization
Virtual network embedding
Speed scaling
Resource allocation
Optimization

ABSTRACT

Network virtualization is key to cloud services, in that it enables multiple users to share a physical infrastructure through abstraction. We propose an online virtual network (VN) embedding scheme which jointly considers load balancing and energy saving so as to maximize the profit of Infrastructure Providers (InPs). For load balancing, we propose to minimize a convex objective which penalizes mapping of VNs to overloaded resources. For energy saving, we consider two popular energy models: speed scaling and power-down. In the speed scaling model, energy consumption is modeled as a convex function of the load imposed on resources. We observe that both load-balancing and energy-saving objectives superadditively penalize high utilization/congestion at resources, and that such synergistic nature of the objectives leads to efficient joint optimization. In the power-down model, a fixed cost exists for keeping a node powered on, which is characterized by a nonconvex energy curve. In this case, we propose an iterative algorithm which explores the trade-offs between load balancing versus cost reduction from power-down of idle servers, in a controlled way. Our algorithm performs a sequential node and link mapping; in particular, for link mapping, we adopt randomized rounding with path stripping in order to obtain a constant factor approximation to the minimum penalty for link utilization. Numerical experiments show the efficacy of our algorithm in servicing VN requests of various topologies and resource requirements.

1. Introduction

The last decades have witnessed tremendous success of the Internet, which ironically has hindered the evolution and innovation of the Internet architecture. Such resistance to change, or *ossification* [1], made it difficult to deploy new technologies and meet diverse service requirements in the existing architecture. Network virtualization [1–4] has gained significant interest so as to overcome the ossification; it allows heterogeneous network services to coexist in a shared physical infrastructure. Network virtualization is a key enabler for cloud computing services which enjoyed widespread success, e.g., Amazon AWS [5], Google Cloud Platform [6], and Microsoft Azure [7]. Other applications include the virtual testbeds for large-scale network experiments, e.g., PlanetLab [8], GENI [9], Emulab [10].

A set of virtualized end-to-end services are provisioned in the form of virtual networks (VNs). A VN consists of virtual nodes and links accompanied by their resource requirements, where we consider CPU and bandwidth as the node and link resources respectively. The Service Providers (SPs) create VNs by leasing resources from an Infrastructure Provider (InP) in order to provide customized computing services to end-users. The InP overlay the requested VNs onto physical infrastructure network which is called the *substrate network (SN)*. In this paper, we investigate the virtual networking embedding (VNE) problem of how the InP efficiently maps VN requests onto the SN subject to resource capacity constraints. Our goal is to maximize the *profit* of the InP, where the profit in our problem has the meaning of revenue minus operation costs. Thus, we would like to increase the revenue and reduce the cost at the same time. Although the VNE problem is extensively studied recently [11–15], our approach is novel in that it considers the operation profit of infrastructure providers via jointly optimizing revenue and cost. Let us define the revenue and cost in more detail as follows.

1) Revenue: Revenue is defined as the payment received by the InP through leasing resources to SPs. We assume that revenue is an increasing function of the amount of mapped resources. Thus, in order to maximize the revenue, the InP would like to accept as many VN requests as possible, whereas a VN request whose resource requirement exceeds the available capacity is rejected. However, it is known that online VNE problem is NP-hard, e.g., even for exactly determining the feasibility of mapping a VN onto a substrate network without violating node and link capacity constraints at the time of embedding [12]. Moreover, in practice, the InP must make online embedding decisions,

* Corresponding author.
E-mail addresses: pyoung1101@korea.ac.kr (C.K. Pyoung), sjbaek@korea.ac.kr (S.J. Baek).

https://doi.org/10.1016/j.comcom.2018.02.004
Received 20 June 2017; Received in revised form 29 January 2018; Accepted 9 February 2018
Available online 21 February 2018
0140-3664/

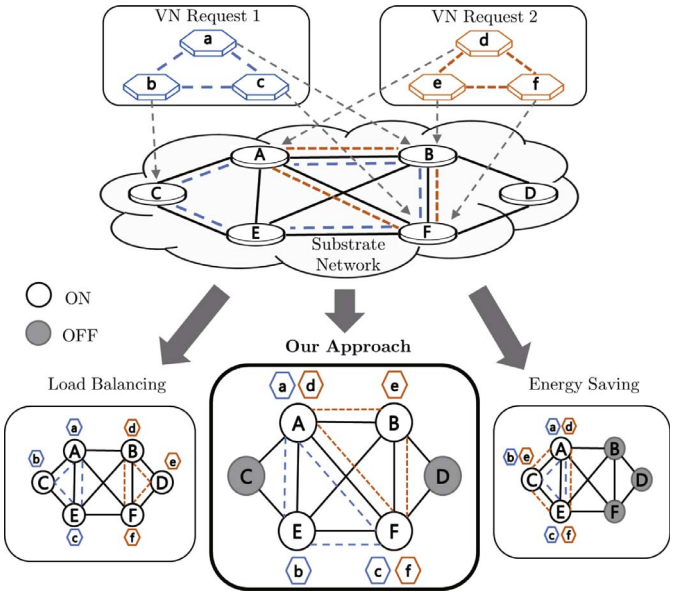

Fig. 1. VNE for joint load balancing and energy saving. The example in the middle shows our approach for power-down model, i.e., it distributes the load from virtual requests over the substrate network, and at the same time, powers down a considerable number of physical nodes. Our scheme is designed to systematically explore the performance tradeoffs between energy efficiency and acceptance ratio (load balancing).

which is even challenging due to stochastic nature of arrivals and durations of VNs whose statistics are typically unavailable at the InP. Many approaches resort to heuristics [11,12] as a best-effort online allocation of resources. Notably, it was found that *load-balancing* in VN mapping is desirable [11,13] for an efficient use of substrate resources. Such approaches attempt to evenly distribute loads across the SN, which in particular increases the likelihood of VN acceptance. To that end, we introduce an *penalty* function for VN mapping which is *convex* in the resource utilization. Specifically the function penalizes VN mapping onto overloaded resources in a superlinear manner, which helps create more room in the SN resources for future VN requests. This increases the likelihood of such requests being accepted, which increases the revenue (Fig. 1).

2) Cost: The main consideration is the energy cost of operating network equipments incurred at the InP. Energy consumption is a big concern for network operators facing the growing demand for cloud computing. The total electricity consumption of data centers in the U.S. is estimated to reach 73 billion kWh/y by 2020 [16]. Consequently, numerous energy-saving techniques have been developed for computing and networking elements. There are two types of widely used techniques: *speed scaling* and *power-down*. Speed scaling dynamically adapts processing speed to system loads so as to save energy. For example, with Dynamic Voltage and Frequency Scaling (DVFS), a system with low loads is run at a slower speed by adjusting voltage and frequency. In speed scaling, power consumption $p(s)$ of a node with load s satisfies $p(s) \propto s^{\alpha}$ where $\alpha > 1$ is a device-specific parameter [17]. The second technique is called power-down, in which the nodes are assumed to have a fixed *start-up* cost $\sigma > 0$, i.e., it takes a fixed power cost σ to keep a node powered on. Unused servers can be put to a power saving mode, e.g., hibernation, in which the servers consume much less power. A number of measurement studies [14,18,19] showed that the actual power consumption of a server is approximately proportional to CPU loads once the server is powered on. In this case, $p(s)$ increases linearly in s at rate μ if $s > 0$.

In summary, speed scaling represents an ideal case in which fine-grained control of speed and power is available, whereas the power-down model reflects more realistic scenario for contemporary servers. We consider both speed scaling and power-down models in this paper, and design energy-efficient schemes separately for each model.

In this paper, we propose efficient online VNE algorithms for profit maximization, where we simultaneously perform load balancing for revenue increase and energy saving for cost reduction. To that end, we choose the aforementioned energy curves and convex penalty as the cost functions (objectives) associated with the VN mapping. We consider a two-phase mapping scheme, where the first phase maps the nodes and the second phase maps the links of requested VN.

In the node mapping, we consider a multi-objective optimization for joint load balancing and energy saving via a Mixed-Integer Linear Programming (MILP) formulation. Since MILP for VNE problems is NP-hard [13], we relax the integrality constraints, and solve a linear program (LP) instead, then round the fractional solution to a mapping decision (integer solution). We consider both power models of speed scaling and power-down, and propose two node-mapping algorithms for different power models. Our key observation is as follows. In case of speed scaling, both load-balancing and energy-saving objectives penalize highly utilized/congested resources in a superadditive manner, and such a synergistic nature of the objectives leads to an efficient joint optimization. In case of power-down, load-balancing is not necessarily good for energy-saving, because it may be more power efficient to assign loads to servers which are already loaded (powered on) to avoid start-up costs. This property results in a nonconvex optimization problem, for which we consider the following heuristic. We propose a two-stage node mapping algorithm. In the first stage, we systematically determine the set of nodes to be powered down so as to achieve a good trade-off between power saving and load balancing. In the second stage, we *rebalance* loads over the active nodes from the first stage in order to achieve a better balance in load distribution so as to improve acceptance ratio. Our approach effectively captures start-up costs in the joint optimization, and explores a trade-off between load-balancing and energy-saving for the power-down model.

In the link mapping, we consider a LP formulation to balance the link stress of the substrate network. In particular, we use *randomized rounding with path stripping* [20] so as to obtain practical routing from a multipath solution resulting from the LP formulation. We show that the link mapping guarantees a constant-factor approximation to the minimum link cost irrespective of network size. We perform extensive simulations under various types of VN requests in terms of their resource requirements and topology. We demonstrate that our method outperforms existing schemes in numerous performance metrics, such as revenue, acceptance ratio, power consumption and profit.

We summarize our contributions as follows.

- We propose a framework for energy-aware VN embedding for profit maximization of infrastructure providers. Joint load-balancing and energy-saving algorithms are proposed for widely used energy models: speed scaling and power-down.
- We introduce penalty functions which are convex, increasing and piecewise linear in the node and link utilization. Load balancing is achieved by penalizing mapping VNs onto congested resources. Moreover, our penalty can approximate any convex cost function including link delays, and thus is quite general.
- For the speed scaling model, we propose an algorithm for node mapping which prevents congestions at resources, which proves be effective for both energy-saving and load-balancing.
- For the power-down model, we propose a two-stage iterative algorithm for node mapping to strike a balance between load balancing and energy saving.
- For link mapping, we propose to use randomized routing with path stripping. We show that, by using the convex properties of penalty functions, our algorithm achieves a constant-factor approximation to the minimum link penalty.
- Through extensive simulations, we show our algorithm outperforms existing schemes in many important metrics.

The remainder of this paper is organized as follows. We review the related work in Section 2. We describe the system model of substrate network in Section 3.1 and the associated VNE problem in Section 3.2. The formulation is introduced in Section 3.3. In Section 3.4, we present the node mapping algorithm with speed scaling. Node mapping for power-down model is proposed in Section 3.5. In Section 3.6, we propose the link mapping scheme. We evaluate the performance of our algorithm via simulations in Section 4. Section 5 concludes the paper.

2. Related work

There exists a considerable volume of research on the VNE problem. We first survey previous studies on VNE in general, then review recent works on the energy-aware VNE schemes.

An efficient management of computer and network resources, due to their large scale, is important for provisioning virtualized services. Prior research has examined various performance criteria, e.g., revenue, costs, acceptance ratio, etc., associated with the mapping of virtual resources [11–13,21–26]. Zhu and Ammar [11] presented a basic VNE algorithm which greedily maps virtual nodes to substrate nodes with low stress (relative resource usage) and interconnects them using the shortest path algorithm. By adopting a selective reconfiguration mechanism, they show that their algorithm is effective for balancing the node and link stress while maintaining reconfiguration costs low. However, they assumed an unlimited availability of network resources. Yu et al. [12] formulated VNE as multi-commodity flow (MCF) problem to find a feasible allocation in polynomial time. The authors proposed to use path splitting via multipath routing and path migration, which are assumed to be supported by the substrate network. Chowdhury et al. [13] considered a two-phase node and link mapping with location constraints on virtual nodes. They formulate a MILP for the augmented substrate network, and use a relaxation-and-rounding technique for the node and link assignments. Papagianni et al. [23] proposed a virtual resource allocation scheme for networked clouds using similar methodology, but with the consideration of the Quality of Service (QoS) requirements of users. We leverage some of the above approaches, e.g., the definition of revenue [12] and two-phase mapping with relaxation [13], however towards a different goal of joint load balancing and energy saving.

Several works have examined the VNE problem from new perspectives. Cheng et al. [22] adopted a Markov Random Walk model to rank a substrate node based on its resource and topological attributes. Candidate node positions for node mapping are selected according to this resource rank. Zhang et al. [24] introduced the idea of opportunistic resource sharing. Their scheme exploits the time-varying nature of resource requirements of VNs, and is able to achieve more efficient resource utilization as opposed to reserving fixed amount of resource throughout the entire lifetime of VNs. Gong et al. [25] introduced the metric called global resource capacity (GRC) which estimates the embedding potential of the substrate nodes, and proposed a greedy load-balancing algorithm based on GRC. Ayoubi et al. [26] proposed a framework for reliable VN embedding with migration which consists of availability-aware resource allocation and reconfiguration. Although these works do not address energy issues, they bring a number of new insights to the design of VNE algorithms. Our penalty-based approach is quite general, in that the penalty function can be calculated for nodes and links connected through arbitrary topology, as long as the information on residual resource is available at the scheduler. Meanwhile, topological information can be further exploited to reduce the load on substrate networks as proposed by Luizelli et al. [27]. The authors studied the impact of different topologies of substrate networks, e.g., ISP and datacenter topologies, for VNE, and proposed algorithms to enhance between rejection rate and resource use depending on the topological properties of substrate networks.

Recently there has been considerable interest on energy-aware allocations of virtual resources. As mentioned earlier, energy saving techniques can be categorized into speed scaling and power-down; we first review strategies based on speed scaling. Andrews et al. [17] presented a framework on the fundamental limits of speed scaling in network routing and optimization. Stanojevic et al. [28] adopted speed scaling to model the power consumption at the CPUs of data center servers, where the CPU adapts its processing speed to current load. They considered an optimal assignment of CPU demands to data center servers, and proposed a distributed algorithm which minimizes the total energy costs. In a different context, the work [29] investigated dynamic resource allocation in cellular networks. The authors propose to adapt the transmission rates of base stations to offered loads so as to achieve a trade-off between energy and delay. Kwak et al. [30] presented a dynamic speed scaling algorithm called SpeedControl which explores the energy-delay trade-off relationship in smart phone applications. SpeedControl jointly optimizes CPU clock and network speed which involves application scheduling, CPU speed control, wireless interface selection, and transmit power control in order to minimize the energy consumption at the CPU and network interface. A similar approach was proposed for mobile cloud systems [31]. The authors proposed a joint optimization algorithm consisting of dynamic cloud offloading policy and CPU/network speed scaling so as to minimize CPU and network energy consumption for given delay constraints. In [32], the authors studied the equilibrium arising from the interaction between load balancing and energy efficiency under speed scaling, in a different context of dispatching jobs to multiple servers in a processor sharing system. They proposed distributed algorithms to reach the equilibrium, and showed that the equilibrium can be inefficient due to system heterogeneity.

A number of VNE schemes have been proposed for the power-down model as well. Su et al. [14] introduced resource *consolidation* which aggregates several VN requests to a smaller number of substrate nodes which are already actively running. The authors showed that consolidation helps to avoid powering inactive nodes on, and proposed an efficient algorithm based on particle swarm optimization. They proposed a consolidation scheme in an earlier work [33], based on node ranking determined from best-fit and worst-fit algorithms associated with the resource availability. Botero et al. considered a consolidation method which minimizes the number of active servers [34], and subsequently extended the method by combining it with load balancing [35]. Nonde et al. [36] applied the consolidation technique to the energy-efficient VNE design for clouds in IP over WDM networks. Sun et al. [37] introduce an energy-aware VN provisioning algorithm which jointly optimizes workload-dependent and workload-independent power consumption in cloud-based data centers. Chen et. al [15] focus on the node mapping for energy minimization, and formulate a linear program to minimize the total CPU energy costs. In a different context of virtual machine (VM) assignments for data centers, Wang et al. [38] considered an energy model which combines speed scaling and power-down; the energy curve has a start-up cost, and thereafter increases superadditively in the system load. The same model has been considered in several other works [17,39]. They considered both VM assignment and traffic engineering to achieve power efficiency, and proposed a VM assignment algorithm which reduces the number of active racks, and distributes a cluster of VMs (called super-VM) to the active racks in a balanced manner.

None of the aforementioned works have covered both speed scaling and power-down models at the same time in association with load balancing. In particular, several works [34,35] have examined energy efficiency of VNE based on mixed integer programming for small size networks. In [34], the authors proposed energy efficient VNE which minimizes the number of active servers. However, the load balancing aspect of mapping was not considered. The work [35] considers both energy saving and load balancing; however, the main purpose of their algorithm is energy efficiency, and load balancing is considered mainly as a means to tie-breaking of allocations which result in the same energy consumption. Unlike these approaches, we use more realistic and

sophisticated energy models, such as speed scaling and power-down, and thoroughly explore the performance tradeoffs between load balancing and energy consumptions. This approach is not present in the previous works. We observe that, when we jointly optimize for load balancing and power saving, different energy models lead to very different strategies. For speed scaling, the objectives are "aligned"; for power-down, they seem to be "orthogonal". A careful optimization capturing these aspects is essential for profit maximization, which we attempt to accomplish in our work.

3. System model

3.1. Substrate network

We model the substrate network as an undirected graph $G^S = (N^S, E^S)$ where the set of vertices and edges, or N^S and E^S, denote the set of substrate nodes and links respectively. Substrate node $n^S \in N^S$ has a CPU capacity denoted by $\text{CPU}(n^S)$, $\text{CPU}{:}N^S \to \mathbb{R}^+$. Link $e^S \in E^S$ has a bandwidth capacity denoted by $\text{BW}(e^S)$, $\text{BW}{:}E^S \to \mathbb{R}^+$. Let $\mathscr{P}^S$ denote the set of all substrate paths in $\mathscr{P}^S$: $N^S \times N^S \to 2^{|E|}$. Let $p^S(u^S, v^S) \subseteq 2^{|E|}$ denote the set of all possible paths from the source node u^S to destination node v^S.

3.2. VN embedding problem

We model a VN request as an undirected graph $G^V = (N^V, E^V)$. The CPU demand of virtual node $n^V \in N^V$ is denoted by $\text{CPU}(n^V)$, $\text{CPU}{:}N^V \to \mathbb{R}^+$. Virtual link $e^V \in E^V$ has a bandwidth demand denoted by $\text{BW}(e^V)$, $\text{BW}{:}E^V \to \mathbb{R}^+$.

Each VN request has a finite duration which we call the lifetime. When a VN request arrives at the substrate network, we decide whether to accept or reject the request. If a VN request is accepted, the resource required for the VN request is allocated on the selected substrate nodes and paths. If a VN request leaves the substrate network, the previously allocated resources are released.

The proposed mapping process is divided into two phases as follows.

- **Node mapping**: The residual CPU capacity of substrate node $n^S \in N^S$ is denoted by $R_N(n^S)$ which represents the available CPU capacity of the substrate node n^S. Each virtual node from a VN request is mapped onto a substrate node, defined by a mapping F_N: $N^V \to N^S$ provided that $\text{CPU}(n^V) \le R_N(F_N(n^V))$. $R_N(n^S)$ can be found by subtracting all the CPU requests by the virtual nodes which are currently mapped onto n^S from the CPU capacity of n^S. We assume that no two virtual nodes can be mapped onto the same substrate node for a given VN request [14,23,37,40].

- **Link mapping**: The residual bandwidth capacity of substrate link $e^S \in E^S$ is denoted by $R_L(e^S)$ which is the total amount of bandwidth available on substrate link e^S. $R_L(e^S)$ is obtained by subtracting the bandwidths of all the virtual links which are currently mapped onto e^S from the bandwidth capacity of e^S. The residual bandwidth capacity of a substrate path $p^S \in \mathscr{P}^S$, denoted by $R_L(p^S)$, is defined as

$$R_L(p^S) = \min_{e^S \in p^S} R_L(e^S).$$

Each virtual link is mapped onto a set of substrate paths by a mapping F_E: $E^V \to \mathscr{P}^S$ where $\text{BW}(e^V) \le R_L(F_E(e^V))$.

Table 1 is a summary of the notations used in this paper.

3.3. Formulation

3.3.1. Performance metrics

We define the following three performance metrics to evaluate our proposed algorithms.

Table 1
Notations of network model.

Notation	Description
G^S	Substrate network graph.
N^S	Set of substrate network nodes.
E^S	Set of substrate network links.
$\mathscr{P}^S$	Set of all substrate paths.
$p^S(u^S, v^S)$	Set of all possible substrate paths between substrate node u^S and v^S.
$\text{CPU}(n^S)$	CPU capacity of substrate node n^S.
$\text{BW}(e^S)$	Bandwidth capacity of substrate link e^S. Also denoted by $\text{BW}(u, v)$ if $e^S = (u, v)$ for some $u, v \in N^S$.
G^V	Virtual network graph.
N^V	Set of virtual nodes.
E^V	Set of virtual links.
$\text{CPU}(n^V)$	CPU demand of virtual node n^V.
$\text{BW}(e^V)$	Bandwidth demand of virtual link e^V.
$R_N(n^S)$	Residual CPU capacity of substrate node n^S.
$R_L(e^S)$	Residual bandwidth capacity of substrate link e^S. Also denoted by $R_L(u, v)$ if $e^S = (u, v)$.
$S_N(n^S)$	Amount of CPU resource in use at substrate node n^S
$S_L(e^S)$	Amount of bandwidth in use at substrate link e^S. Also denoted by $S_L(u, v)$ if $e^S = (u, v)$.

1) **Revenue**: Revenue is defined as the income generated by hosting a VN request. This is similar to the definition in previous works [12,13]. We assume that the revenue is proportional to the amount of resource required to map VN G^V onto SN. Specifically, the revenue earned by mapping G^V is given by

$$\sum_{n^V \in N^V} \text{CPU}(n^V) + \sum_{e^V \in E^V} \text{BW}(e^V). \tag{1}$$

2) **Cost**: There exist operational costs incurred at InP associated with serving a VN request. We consider the power cost which is defined in the following section.
3) **Profit**: Profit is defined to be revenue minus costs. Profit maximization is the eventual goal of the InP, hence we would like to maximize the revenue and minimize the cost. However, the exact maximization of revenue, especially for large substrate networks, is hard; we hence attempt to minimize the penalty and costs where the penalty serves as an indirect measure of the revenue.

3.3.2. Objectives

We introduce the following objective functions to formulate the profit maximization problem.

- **Penalty**:
  The VNE problem is NP-hard even as an offline version [12], i.e., with the complete knowledge of arrival/departure times and resource requirements of VNs. Meanwhile, simulation studies in [11,13] revealed that load balancing enables an efficient use of substrate resources and enhances the acceptance ratio, which lead to increased revenue. Thus, instead of directly maximizing the revenue, we consider a heuristic of minimizing the *penalty* for the purpose of load balancing. Our penalty function (defined in Section 3.3.3) penalize mapping VNs onto heavily loaded substrate nodes/links; we set the penalty as an objective function in our optimization.
- **Cost**: In order to energy saving, we consider two power models for speed scaling and power-down, respectively. The power cost for each model is defined in the following section.

We would like to find the mapping of given VN which jointly minimizes penalty and cost at the arrival of the VN request. This results in a multi-objective optimization, and we specifically define these objectives as follows.

3.3.3. Penalty

We would like to admit as many VNs as possible so as to increase the revenue. Because a VN is admitted only if there exist available resources, we will focus on *load balancing* to avoid "congestion" of substrate resources, i.e., to avoid mapping a VN onto heavily loaded nodes and links. By definition, the revenue is proportional to the amount of mapped resource (e.g., CPU loads and link bandwidth). This definition is reasonable because infrastructure provider typically charges the costs proportional to its resource usage to the users. However, the revenue itself is oblivious of resource congestion. Thus, resource mapping based only on revenue functions can lead to highly congested resources, which will prevent future VNs from using such resources. Such load balancing can be done by *penalizing* the substrate resource by its utilization. To that end, we introduce a penalty function which is a *convex* increasing function of the utilization of nodes and links.

The idea of penalizing resource congestion with convex functions is motivated by the seminal work [41] by Fortz et al. The work concerns setting link weights for QoS-aware routing of Internet traffic. The authors propose to penalize links by a piecewise linear convex function of the link utilization, which is shown to improve load balancing and resource utilization. The function they use is an approximation of $\gamma(f) := f/(C-f)$ where f and C denote the bandwidth usage and capacity respectively. Note that $\gamma(f)$ represents delay in queuing theoretic models. The authors however argued that the exact form of penalty functions is less important, but it is the *superadditive* property of penalty which contributes to good load balancing [41]. Recently, the convex penalty approach was also made in [42] where the authors proposed a congestion-aware flow mapping for data center networks. The algorithm is based on a cost function which is convex and increasing in link utilization. Their algorithm is shown to have a good load balancing performance, e.g., it achieves asymptotically minimum link congestion costs.

We define the penalty function for CPU utilization at a node as follows:

$$\Gamma_{\mathrm{C}}(\beta) = c_{l_\beta+1}(\beta - J_{l_\beta}) + \sum_{q=1}^{l_\beta} c_q(J_q - J_{q-1}) \tag{2}$$

where $l_\beta := \max\{k | J_k \le \beta\}$, $\beta \in [0, 1]$ denotes the CPU utilization, and J_{l_β} is some constant for all $1 \le l_\beta \le N_{\mathrm{C}}$ where N_{C} denotes the number of slopes, and $c_1 \le c_2 \le \cdots \le c_{N_{\mathrm{C}}}$ denote the increasing slopes. $\Gamma_{\mathrm{C}}(\beta)$ is a convex and increasing piecewise linear function as shown in Fig. 2. Similarly, we introduce the penalty function for bandwidth utilization as follows:

$$\Gamma_{\mathrm{L}}(\beta) = a_{l_\beta+1}(\beta - M_{l_\beta}) + \sum_{p=1}^{l_\beta} a_p(M_p - M_{p-1}) \tag{3}$$

where $l_\beta := \max\{k | M_k \le \beta\}$, $\beta \in [0, 1]$ denotes the bandwidth utilization at the link, M_{l_β} is a constant for $1 \le l_\beta \le N_{\mathrm{L}}$ where N_{L} denotes the number of slopes, and $a_1 \le a_2 \le \cdots \le a_{N_{\mathrm{L}}}$ denote the slopes. Penalty functions Γ_C and Γ_L will be included in the objective function of optimization at the time of mapping. Thus, from the above definitions, our method avoids mapping virtual requests onto highly congested substrate resources.

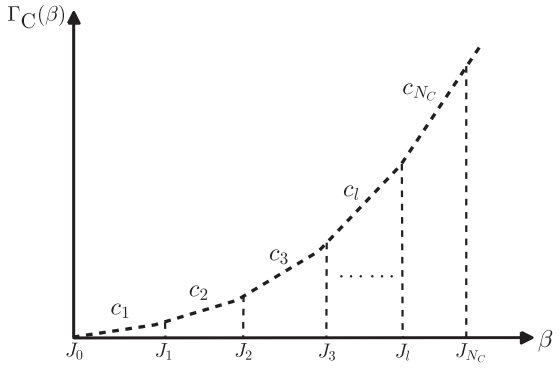

Fig. 2. Penalty function for CPU utilization.

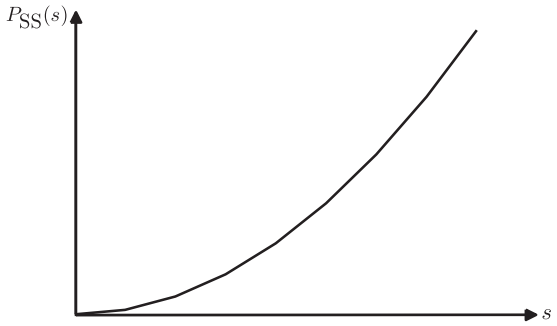

Fig. 3. Example of $P_{\mathrm{SS}}(s)$.

3.3.4. Power cost

We consider two popular models for energy saving: speed scaling and power-down. A separate VNE algorithm is proposed for each model.

3.3.4.1. Speed scaling. Speed scaling techniques enable power saving by running the underutilized system in a reduced speed [17,43–45]. In this model, we assume that the speed scaling technique is adopted for the substrate nodes. The power consumption under speed scaling increases *superadditively* in the load on the substrate node [46,47]. Specifically, we define the power cost function at a substrate node in terms of its CPU loads as follows:

$$P_{\mathrm{SS}}(s) = \mu s^{\alpha} \tag{4}$$

where s denotes the amount of CPU loads at the substrate node, and μ and $\alpha > 1$ are device-specific parameters. Fig. 3 shows an example of $P_{\mathrm{SS}}(s)$.

3.3.4.2. Power-down. In a number of measurement studies on power consumption of data centers [18,19,36], it was found that (i) there exists a considerable start-up cost in powering a server on; (ii) power cost increases approximately linearly in CPU loads. The energy curve under power-down model is defined as follows.

$$P_{\mathrm{PD}}(s) = \begin{cases} \sigma + \mu s, & s > 0, \\ 0, & s = 0. \end{cases} \tag{5}$$

Fig. 4 shows an example of $P_{\mathrm{PD}}(s)$. In practice, σ may account for a significant portion of the power consumption of a server in full utilization [18,19,36]. Thus, substantial power saving is possible by controlling the number of active servers.

Similar to [37], we assume that a substrate node consists of a server

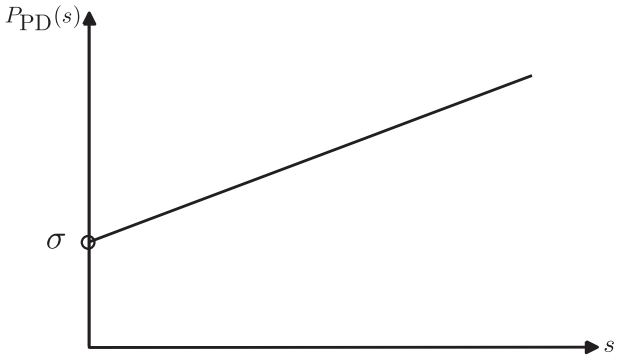

Fig. 4. Example of $P_{\mathrm{PD}}(s)$.

(or a server cluster) and a network equipment (e.g., a switch) where the former provides computing resources (CPU), and the latter provides communication resources. We assume that the server of a substrate node can be powered down, but the network equipment of a substrate node is assumed to be active at all times. Previous works using the power-down model [14,15,34] assumed both nodes and links can be powered down. However, in practice, network equipments may exchange control packets frequently even without data traffic. Thus, we believe that it is practical to keep network modules and interfaces active at all times. In this paper, when we say a node is powered down, it means that only its CPU server is powered down, however its network equipment and the substrate links connected to it remain active.[1] Consequently, we consider the power consumption of only substrate nodes, but not of substrate links, under both power models. Thus, the power cost minimization is relevant only to the node mapping phase, but not to the link mapping phase.

3.3.5. Multi-objective optimization

There are multiple objectives which we would like to minimize simultaneously, i.e., the penalty functions Γ_C in addition to power cost functions $P_{SS}(\cdot)$. Thus, we have multi-objective optimization for which we use a linear scalarization method [50]. Specifically, we consider a single scalarized objective which is a nonnegatively weighted linear combination of the objectives. For example, under speed scaling model, we attempt to minimize

$$\rho_C \Gamma_C + \rho_P P_{SS}$$

where ρ_C and ρ_P are nonnegative weights for scalarization. These constants are chosen according to relative importances of the objectives.

3.4. Node mapping with speed scaling

3.4.1. Pre-selection of substrate node

We first pre-select a set of substrate nodes for mapping, called candidate nodes. Candidate nodes are selected as follows. We assume that each virtual node is associated with its geographic location in the substrate network [13,51]. For given substrate node u and virtual node v, let $\|u - v\|$ denote the distance between physical locations of u and v. A virtual node can be mapped onto substrate nodes which are within distance D^v. Distance can be translated to delay or the minimum number of hops between two node locations. In addition, a candidate substrate node must have sufficient CPU resources. Namely v can be mapped onto only a node in $\Phi(v) \subseteq N^S$ where

$$\Phi(v): = \{u \in N^S | \|u - v\| \le D^v, \ \ \mathrm{CPU}(v) \le R_N(u)\}.$$

The distance-based pre-selection of nodes is used in other works as well [13,51].

3.4.2. Problem formulation

Node mapping with speed scaling is obtained by the solution to following MILP which optimizes for joint load balancing and energy saving.

NM − SS

$$\text{minimize} \quad \rho_P \sum_{v \in N^S} \left[P_{SS} \left(S_N(v) + \sum_{u \in N^V} \mathrm{CPU}(u) x_{uv} \right) \right] \tag{6}$$

$$+ \rho_C \sum_{v \in N^S} \left[\Gamma_C \left(\frac{S_N(v) + \sum_{u \in N^V} \mathrm{CPU}(u) x_{uv}}{\mathrm{CPU}(v)} \right) \right] \tag{7}$$

subject to

$$x_{uv} \in \{0, 1\}, \quad \forall\, u \in N^V, \forall\, v \in N^S. \tag{8}$$

$$f^i_{wv}, f^i_{vw} \ge 0, \quad \forall\, (w, v) \in E^S, \ \ i = 1, \cdots, |E^V| \tag{9}$$

$$\sum_i |f^i_{wv} - f^i_{vw}| \le R_L(w, v), \quad \forall\, (w, v) \in E^S, \tag{10}$$

$$R_N(v) \ge \mathrm{CPU}(u) x_{uv}, \quad \forall\, u \in N^V, \forall\, v \in N^S, \tag{11}$$

$$\sum_{u \in N^V} x_{uv} \le 1, \quad \forall\, v \in N^S, \tag{12}$$

$$\sum_{v \in N^S} x_{uv} = 1, \quad \forall\, u \in N^V, \tag{13}$$

$$\sum_{w \in N^S} (f^i_{vw} - f^i_{wv}) = \mathrm{BW}(e^V_i) \left[x_{s_i v} - x_{t_i v} \right],$$
$$\forall\, i, \ \forall\, s_i, t_i \in N^V, \ \forall\, (w, v) \in E^S,$$
$$\text{variables:} \ \ f^i_{wv}, \ \ x_{uv}. \tag{14}$$

The formulation is akin to the multicommodity flow problem (MFP) in [13], however the control variables are different as follows. $x_{uv} \in \{0, 1\}$ denotes the binary mapping variable which is 1 if virtual node u is mapped onto substrate node v. The bandwidth requirement of virtual link $e^V_i = (s_i, t_i) \in E^V$ is regarded as a directed flow, where the direction is arbitrarily assigned, with size $\mathrm{BW}(e^V_i)$ where virtual nodes s_i and t_i are set to source and sink nodes respectively. Flow variable f^i_{uv} represents the amount of flow from u to v on substrate link (u, v) contributed by the ith virtual link ($1 \le i \le |E^V|$). Note that f^i_{uv} only accounts for the amount of ith virtual flow in the direction from u to v, and thus is always nonnegative. If the direction of ith virtual flow on substrate edge (u, v) is from v to u, we have $f^i_{uv} = 0$. From the above definition, $f^i_{wv} - f^i_{vw}$ has the following interpretation; its absolute value $|f^i_{wv} - f^i_{vw}|$ is the magnitude of the flow; its sign represents the direction, i.e., it is positive if the flow direction is from w to v, and negative otherwise. Constraint (10) accordingly states that the total magnitude of virtual flows mapped on substrate edge (w, u) cannot exceed the residual bandwidth $R_L(w, v)$.

In (6), the objective function represents the total power cost associated with CPU usage at substrate nodes; (7) represents the penalty associated with CPU utilization.

(10) and (11) are constraints on the resource capacity at substrate nodes and links. Constraint (12) ensures that at most one virtual node can be mapped onto a substrate node. Constraint (13) means that one of substrate nodes is selected for each virtual node.

Constraint (14) represents the flow conservation equation. The flow of amount $\mathrm{BW}(e^V_i)$ is associated with source node s_i and sink node t_i. The LHS of (14) is the total outflow from substrate node u contributed by ith virtual link. If $x_{s_i u}$ is equal to 1, meaning that source (virtual) node is mapped to u, the RHS of (14) becomes a positive demand given by $\mathrm{BW}(e^V_i)$. Similarly, if $x_{t_i u}$ is equal to 1, constraint (14) has a negative demand $-\mathrm{BW}(e^V_i)$. Otherwise, the RHS of constraint (14) is zero, hence inflow and outflow from ith virtual link are balanced at node u.

NM-SS is MILP and is hard to solve exactly, mainly due to integral constraints, e.g., determining the feasibility of a VN mapping is reducible to an NP-hard problem [12]. Thus, we relax x_{uv} as a continuous variable in [0, 1]. In this case, **NM-SS** is convex optimization, and thus can be solved in polynomial time. After solving the relaxed version of **NM-SS**, we round the fractional values of x_{uv} in order to obtain integral solutions.

Observations: The power cost function in (6) and penalty function (7) of CPU utilization are both superadditive functions in the resource usage. This implies that load balancing is helpful for power saving as well. Thus, our node mapping tend to map virtual nodes to "lightly loaded" substrate nodes in order for load balancing, which contributes further to power saving.

[1] Once a network module is powered on, its power consumption does not change much with network traffic [33,48,49]. Thus, we assume that the network equipment of a substrate node consumes constant power at all times.

Input : VN request $G^V = (N^V, E^V)$
Output : Mapping F_N
/* 1. Pre-selection of substrate nodes */
for each $v \in N^V$ **do**
  $\Phi(v) \leftarrow \left\{u \in N^S \,|\, \|u - v\| \leq D^v,\ \text{CPU}(v) \leq R_N(u)\right\}$
  if $\Phi(v) = \emptyset$ **then**
    Reject request G^V
    break
  end if
end for
/* 2. Solve relaxed MILP */
Solve relaxed **NM-SS** with additional constraint on pre-selection such that, for each $v \in N^S$, let $x_{uv} = 0$ for all $u \in N^V$ such that $u \notin \Phi(v)$. Reject request G^V if **NM-SS** is infeasible.
/* 3. Node mapping by rounding */
$\phi \leftarrow \emptyset$
for $i = 0; i < |N^V|; i$++ **do**
  /* prioritize the virtual nodes with large CPU demands */
  Select $u \in N^V$ with the maximum CPU(u) among unmapped virtual nodes.
  if $\Phi(u) \backslash \phi = \emptyset$ **then**
    Reject the request G^V
    break
  else
    $F_N(u) \leftarrow \arg\max\{x_{uz} | z \in \Phi(u) \backslash \phi\}$
    $\phi \leftarrow \phi \cup \{F_N(u)\}$
  end if
end for

Algorithm 1. Node mapping with speed scaling.

3.4.3. Algorithm

The node mapping algorithm with speed scaling is outlined in Algorithm 1. In summary, we first pre-select a set of substrate nodes for mapping from Step 4 to Step 10. A virtual node is mapped onto one of its candidate nodes. Next, we solve relaxed **NM-SS** in Step 12. If **NM-SS** is feasible, we then perform rounding of the fractional solutions of relaxed **NM-SS** to mapping decisions from Step 14 to Step 25.

Finally, we perform node mapping by rounding the mapping solutions of the relaxed LP. In doing so, we sort the virtual nodes by its CPU demand, and map the virtual nodes in the descending order of their demands; see Step 17. The idea is to improve acceptance ratio; for substrate node v, there can be multiple virtual node u such that x_{uv} is maximum, i.e., multiple virtual nodes may "compete" for v. Virtual nodes with higher CPU demands are more likely to cause rejection of the associated VN request. Thus, nodes with higher demands are given higher priority in our mapping. Finally, for given u, if there are multiple v which achieves the maximum x_{uv}, we use random tie-breaking.

3.5. Node mapping with power-down

Node mapping strategies with power-down differ substantially from those with speed scaling. Unlike speed scaling, load balancing does not necessarily help reducing power costs due to the presence of start-up costs. For example, if load balancing is emphasized in node mapping, inactive nodes are prioritized for mapping because they have zero load. However, such mapping may incur substantial power costs by activating many inactive nodes. It is more power efficient to perform mapping onto already active substrate nodes, which however may result in poor load balancing. Hence, power efficiency and load balancing are in a trade-off relationship.

Our goal is to explore this trade-off in a controlled way; to that end, we propose a two-stage algorithm for node mapping; in the first stage, we determine the number of nodes to be activated by iteratively solving parameterized MILP; in the second stage, we distribute loads over the active nodes determined from the first stage, which we call *rebalancing*.

3.5.1. Stage 1: determining the set of active substrate nodes

The purpose of Stage 1 is to determine the set of substrate nodes to be powered on. We first formulate an optimization similar to **NM-SS** by replacing power function $P_{\mathrm{SS}}(\cdot)$ in the objective by $P_{\mathrm{PD}}(\cdot)$ in (6). Denote the set of active and inactive nodes prior to mapping by N^S_{on} and N^S_{off} respectively. We further simplify the total power cost P_{SS} in the objective as follows (also see Theorem 1 of [33]).

$$\sum_{v\in N^S}\left[P_{\mathrm{PD}}\left(S_N(v)+\sum_{u\in N^V}\mathrm{CPU}(u)x_{uv}\right)\right] \tag{15}$$

$$\begin{aligned}
&=\sum_{v\in N^S_{\mathrm{on}}}\left[P_{\mathrm{PD}}\left(S_N(v)+\sum_{u\in N^V}\mathrm{CPU}(u)x_{uv}\right)\right]\\
&+\sum_{v\in N^S_{\mathrm{off}}}\left[P_{\mathrm{PD}}\left(\sum_{u\in N^V}\mathrm{CPU}(u)x_{uv}\right)\right]\\
&=\sigma|N^S_{\mathrm{on}}|+\mu\sum_{v\in N^S_{\mathrm{on}}}\left\{S_N(v)+\sum_{u\in N^V}\mathrm{CPU}(u)x_{uv}\right\}\\
&+\sigma\sum_{v\in N^S_{\mathrm{off}}}\sum_{u\in N^V}\mathbf{1}(x_{uv}>0)+\mu\sum_{v\in N^S_{\mathrm{off}}}\sum_{u\in N^V}\mathrm{CPU}(u)x_{uv}\\
&=\sigma\sum_{v\in N^S_{\mathrm{off}}}\sum_{u\in N^V}x_{uv}+(\mathrm{const})
\end{aligned} \tag{16}$$

where (16) is due to $x_{uv}\in\{0,1\}$. The first term of (16) is proportional to the number of virtual nodes mapped onto inactive substrate nodes. Thus, the power cost after the node mapping depends only on the number of nodes which need to be newly activated for mapping.

Similar to **NM-SS**, the objective function has the form $\rho_{\mathrm{P}}P_{\mathrm{SS}}+\rho_{\mathrm{C}}\Gamma_{\mathrm{C}}$ which however is a mixture of conflicting objectives. In order to control the balance between power consumption and resource congestion, we weight the power term of the objective by control parameter $\tau>0$. We formulate the parameterized MILP called **NM-PD**(τ) for node mapping:

$$\begin{aligned}
&\mathbf{NM-PD}(\tau)\\
&\text{minimize}\quad \tau\left[\rho_P\cdot\sigma\sum_{u\in N^V}\sum_{v\in N^S_{\mathrm{off}}}x_{uv}\right]\\
&\qquad+\rho_{\mathrm{C}}\sum_{v\in N^S}\left[\Gamma_{\mathrm{C}}\left(\frac{S_N(v)+\sum_{u\in N^V}\mathrm{CPU}(u)x_{uv}}{\mathrm{CPU}(v)}\right)\right]\\
&\text{subject to}\\
&\qquad(8)-(14)\\
&\text{variables:}\ \ f^i_{wv},\ \ x_{uv}.
\end{aligned} \tag{17}$$

(18)

Note that power cost (15) is replaced by (17) in the objective due to (16). By adjusting parameter τ, one can achieve various trade-off points between power saving and load balancing as follows. From (17), τ serves as a penalty to the powering-on of inactive servers. For larger τ, the solution to **NM-PD**(τ) tend to have smaller value of x_{uv}'s to inactive nodes. This implies that a relatively small number of inactive nodes are activated, thus mapping is inclined towards power saving. By contrast, for smaller τ, we put more emphasis on load-balancing; in the extreme case of $\tau=0$, we have a pure load-balanced mapping due to the convexity of Γ_C. In the sequel we introduce an algorithm to find an intermediate value of τ between two extreme cases; $\tau=0$ (power-oblivious) and $\tau=\infty$ (maximal power saving).

An illustrated example is given in the Stage 1 part of Fig. 5. The substrate network consists of 4 nodes. Two nodes are active, and the other two nodes are inactive. When a VN request with two virtual nodes arrives, maximal load balancing ($\tau=0$) maps the virtual nodes to inactive substrate nodes. By contrast, maximal power saving ($\tau=\infty$) performs mapping on already active nodes so as to minimize the start-

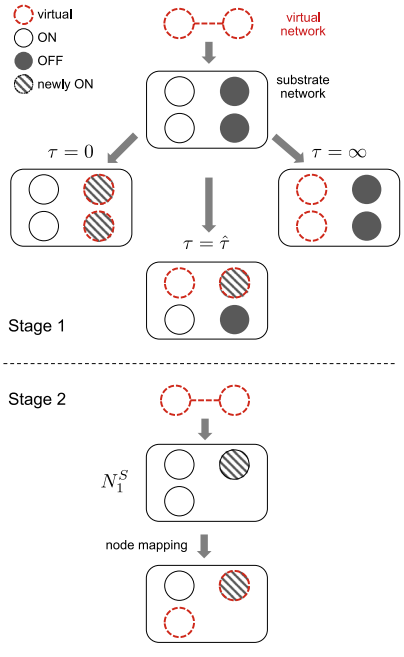

Fig. 5. Node mapping with power-down. In Stage 1, the algorithm determines active node set N^S_1 which consists of the already active and newly activated nodes, controlled by parameter $\hat{\tau}$. In this example, N^S_1 has three nodes; two already active nodes and one newly activated node. Node mapping is actually performed in Stage 2; virtual nodes are mapped onto candidate set N^S_1 in a load-balancing manner. Note that the substrate nodes selected for determining N^S_1 in Stage 1 and the actually mapped nodes in Stage 2 may be different, as shown in this example.

up costs. An intermediate τ ($\tau = \hat{\tau}$) may strike a balance between load balancing and power saving.

Algorithm. We first solve **NM-PD**($\tau_{\max}$) for some large $\tau_{\max} > 0$ in order to approximate the case $\tau = \infty$. Then we run a mapping algorithm similar to Algorithm 1, where in Step 12 we solve **NM-PD**($\tau_{\max}$) instead. After output mapping function $F_N(\cdot)$ is found from the mapping algorithm, we estimate the total power cost which would have incurred if the mapping were performed according to $F_N(\cdot)$; denote the output power cost by P^*. Next we repeat a similar procedure, but in this case we solve **NM-PD**(0) for maximal load balancing (power-oblivious). Denote the output power cost by νP^* for some $\nu \geq 1$. Our goal is to find τ which yields power cost zP^* for some intermediate value z such that $1 \leq z \leq \nu$ so as to balance power consumption and resource congestion. There may be several ways to select z, e.g., in our case we simply set $z := (1 - \theta) + \theta\nu$ for some fixed $\theta \in [0, 1]$ in Step 14. We call zP^* the *target* power cost. For example, if we let $\theta = 0.4$, the target cost is set to 40% point over the range of achievable power costs. This enables us to achieve a target trade-off point between power saving and load balancing.

We search for the value of τ which yields power cost zP^* using bisection. The algorithm is outlined in the Stage 1 part of Algorithm 2. The algorithm may not find τ which results in the *exact* target cost zP^*, because the output power cost after mapping takes discrete values as a function of τ, because the power costs depend only on the number of activated servers. Fig. 6 shows an example of the output power cost as a function of τ. Nevertheless, the bisection in Algorithm 2 is designed such that the eventual cost does not exceed zP^*.

Let $\hat{\tau}$ denote the value of τ after the bisection search is completed. Suppose we run the mapping algorithm with **NM-PD**($\hat{\tau}$), and denote the output mapping rule by $\widehat{F}_N: N^V \to N^S$. Let N^S_{new} denote the set of inactive substrate nodes which would be *newly* activated, if we were to perform mapping according to $\widehat{F}_N$. Consider set N^1_S such that

$$N^S_1 := N^S_{\text{on}} \cup N^S_{\text{new}} \tag{19}$$

where as previously, N^S_{on} is the set of active substrate nodes prior to mapping. Thus, if we were to perform mapping $\widehat{F}_N$, N^S_1 would be the set of active nodes after the mapping. We regard N^S_1 as the candidate set for node mapping in Stage 2. The actual node mapping is deferred to the rebalancing stage during which we perform load-balanced mapping onto N^S_1. This is illustrated in the example in the Stage 1 part of Fig. 5. In the case $\tau = \hat{\tau}$, two active nodes (N^S_{on}) and one newly activated node (N^S_{new}) comprise N^S_1.

3.5.2. Stage 2: rebalance of loads

In Stage 1, we found a set of substrate nodes to be mapped onto and thus to be activated, i.e., N^S_1. The goal of Stage 2 is to perform actual node mapping in a balanced manner onto N^S_1. The motivation is as follows: once the set of nodes to be activated is fixed, the power cost is not change irrespective of how the nodes are mapped onto the set of active nodes, due to (16). Thus, it is always advantageous to perform optimized mapping *after* the set of active nodes is determined, which improves load balancing without incurring further power costs than the target power cost from Stage 1.

Algorithm. The rebalancing stage proceeds as follows; we perform node mapping according to Algorithm 1 with solving a relaxed MILP called **NM-RB** such that (i) we minimize only the congestion penalty term $\rho_C \Gamma_C$ for load balancing; (ii) the candidate node set for the mapping is constrained to N^S_1. **NM-RB** is defined as follows.

NM − RB

$$\text{minimize} \quad \sum_{v \in N^S_1} \left[\Gamma_C \left(\frac{S_N(v) + \sum_{u \in N^V} \text{CPU}(u) x_{uv}}{\text{CPU}(v)} \right) \right]$$

subject to (8) − (14)

variables: f^i_{wv}, x_{uv}.

Unlike **NM-PD**, **NM-RB** focuses only on load balancing, i.e., does not have power cost term in the objective. Rebalancing is described in Stage 2 of Algorithm 2. In conclusion, the output power cost of our algorithm is at most zP^*; hence we do not violate the guarantee of target power from Stage 1. We potentially achieve a better balance in node mapping in Stage 2, since there is no extra power cost involved in rebalancing but only load balancing is considered, which is a salient feature of our algorithm.

An illustrated example of rebalancing is given in the Stage 2 part of Fig. 5. The substrate nodes which were selected for determining N^S_1 in Stage 1 may not be identical to the substrate nodes which have been actually mapped in Stage 2. This is because, different optimization problems are solved in different stages. In Stage 1, **NM-PD**($\hat{\tau}$) is solved for hypothetical mapping, whereas the actual mapping in Stage 2 is based on **NM-RB** in which power costs are not included.

3.6. Link mapping

In this section, we discuss link mapping, i.e., how a virtual link is mapped onto substrate links.[2] As previously, we would like to map the flows from a virtual link onto substrate links in a balanced manner. To that end, we formulate a MFP called **LM** so as to minimize the penalty associated with link congestion. The MFP may yield a fractional solution, in which case one needs multiple paths between virtual nodes in order to allocate flows according to the solution. Such multipath routing is typically undesirable for implementation due to, e.g., packet reordering problem for TCP. In practice, it is preferable to assign a single route for each virtual link. However, finding integer flow solutions to MFP is NP-complete [20]. Thus, as an approximation, we use a randomized rounding algorithm with path stripping [20]. We show that the link mapping algorithm can approximate the true minimum penalty to a constant factor irrespective of network size.

3.6.1. Algorithm

We consider the following MFP.

$$\textbf{LM} \quad \text{minimize} \sum_{(u,v) \in E^S} \Gamma_L \left(\frac{S_L(u,v) + \sum_i |f^i_{uv} - f^i_{vu}|}{\text{BW}(u,v)} \right) \tag{20}$$

subject to

$$\sum_i |f^i_{uv} - f^i_{vu}| \leq R_L(u,v),$$

$$\sum_{u \in N^S} (f^i_{vu} - f^i_{uv}) = \text{BW}(e^V_i)[\mathbf{1}(F_N(s_i) = v) - \mathbf{1}(F_N(t_i) = v)], \tag{21}$$

$$f^i_{uv}, f^i_{vu} \geq 0, \quad \forall\ (u,v) \in E^S,\ s_i, t_i \in N^V,\ \forall\ i = 1, \cdots, |E^V|,$$

variables: f^i_{uv}. (22)

The objective function in (20) is given by the penalty function associated with link utilization. In the constraints on flow conservation given by (21), s_i and t_i denote the virtual source and sink nodes as defined in (14). $F_N(\cdot)$ in (21) is the output mapping function from the node mapping phase. By solving **LM**, we may obtain fractional solutions, i.e., non-integer flows. We use randomized rounding with path stripping [20] in order to obtain a near-optimal solution yielding a single route per virtual link.

The algorithm for mapping ith virtual link, $i = 1, \cdots, |E^V|$, is outlined in Algorithm 3. We first solve **LM** to obtain the optimal fractional flows. With some abuse of notation, we use f^i_e to denote the solution for ith virtual link on substrate link $e \in E^S$; in other words, if $e = (u, v)$, f^i_e represents f^i_{uv} in **LM**. For ith virtual link $e_i = (s_i, t_i)$, let $\bar{s}_i$ and $\bar{t}_i$ denote the

[2] As noted earlier, the same link mapping algorithm is used for both of the power models.

1: Input : VN request $G^V = (N^V, E^V)$
2: Output : Mapping F_N
3: Given: tolerance $\delta \in [0, 1)$, parameters $\theta \in [0, 1]$, $\tau_{\max}$, $i_{\max}$
4: Let N^S_{on} and N^S_{off} as the set of active and inactive nodes respectively
5: **if** $N^S_{\text{off}} = \emptyset$ **then**
6: `/* all nodes are active */`
7: $G^S_{\text{on}} \leftarrow G^S$, **goto** step 35
8: **end if**
9: `/* Stage 1: power-aware load balancing; determine the set of nodes to be powered on */`
10: $\tau \leftarrow \tau_{\max}$ `/* case for maximal power saving */`
11: Run Algorithm 1, where in Step 12 solve **NM-PD**(τ) instead. Denote the power cost after mapping by P^*.
12: $\tau \leftarrow 0$ `/* case for maximal load balancing */`
13: Run Algorithm 1, where in Step 12 solve **NM-PD**(τ) instead. Denote the power cost after mapping by νP^* for some $\nu \geq 1$.
14: $z \leftarrow (1 - \theta) + \theta\nu$ `/* target power cost is` zP^* `*/`
15: `/* bisection */`
16: $a \leftarrow 0$, $b \leftarrow \tau_{\max}$
17: $\hat{\tau} \leftarrow 0, \hat{P} \leftarrow P^*$
18: **for** ($i = 0$; $i < i_{\max}$; i++) **do**
19: $t \leftarrow (a + b)/2$
20: Run Algorithm 1, where in Step 12 solve **NM-PD**(t) instead. Denote the power cost after mapping by Q.
21: **if** $Q \leq zP^*$ **then**
22: $b \leftarrow t$
23: **else**
24: $a \leftarrow t$
25: **end if**
26: **if** $\hat{P} \leq Q \leq zP^*$ **then**
27: $\hat{P} \leftarrow Q$, $\hat{\tau} \leftarrow t$
28: **end if**
29: **if** $(1 - \delta)zP^* \leq Q \leq zP^*$ **then**
30: **break**
31: **end if**
32: **end for**
33: `/* Stage 2: rebalance */`
34: Let N^S_{new} denote the set of newly activated substrate nodes from Stage 1. Let $N^S_1 := N^S_{\text{on}} \cup N^S_{\text{new}}$
35: Perform node mapping by running Algorithm 1 with following changes: (i) replace $\Phi(v)$ by $\Phi(v) \cap N^S_1$; (ii) solve relaxed **(NM-RB)** in Step 12 instead.
36: Power down the nodes with zero load.

Algorithm 2. Node mapping for power-down.

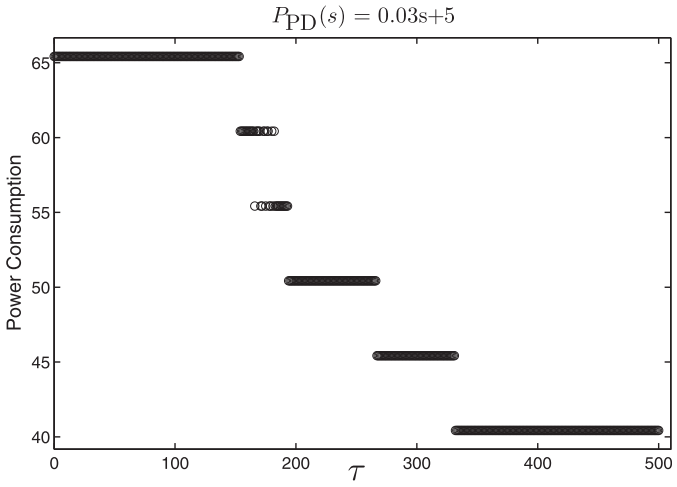

Fig. 6. Output power Q in Step 20 in Algorithm 2 as a function of τ.

substrate nodes such that s_i and t_i are respectively mapped onto $\bar{s}_i$ and $\bar{t}_i$ during the node mapping phase. We would like to route directed flow $\mathrm{BW}(e_i)$ based on the fractional solution from $\bar{s}_i$ to $\bar{t}_i$. To that end, we first perform path stripping as follows. We find a path between $\bar{s}_i$ and $\bar{t}_i$ where the path is assigned with weight ϕ_k that is proportional to the minimum flow on that path. We store that path in Λ, reduce the flow on that path by ϕ_k, and remove any links with zero flow. This procedure is repeated until all the links are removed. Next we perform randomized rounding by randomly selecting a path from Λ with probability proportional to the weight.

3.6.2. Bounds on link penalty

In this section, we analyze the performance of link mapping with randomized rounding in terms of link penalty. We show that the expected penalty under the algorithm is at most constant times the minimum link penalty irrespective of network size. In proving our results, we leverage the analysis in [17] in which the authors derived a cost bound associated with randomized rounding. They considered a polynomial cost function of the form x^{α}. By contrast, our link penalty function Γ_{L} is piecewise linear, and can be considered as more general, because it can approximate arbitrary increasing convex function by letting Γ_{L} be the point-wise maximum of a sufficient number of linear functions.

Let $d_i := \mathrm{BW}(e_i)$, $e_i \in E^V$, denote the bandwidth demand for ith virtual link. We assume that d_i's are integer demands. We first consider the case of uniform demands, i.e., $d_i = d$, $\forall\, i$.

Proposition 1. *Suppose $d_i = d$, $\forall\, i = 1, \cdots, |E^V|$. The expected cost obtained from the randomized rounding is at most γ times the minimum link penalty, where γ is a constant.*

Proof. The proof is given in Appendix A. □

The result can be generalized to the case where d_i differ among virtual links $i = 1, \cdots, |E^V|$.

Proposition 2. *For general demands, the expected cost of link mapping is at most constant times the minimum penalty.*

Proof. The proof is given in Appendix B. □

4. Performance evaluation

In this section, we evaluate the performance of our algorithm through simulation. Our simulation focuses on evaluating profit, revenue, acceptance ratio and power consumption. These performance metrics are measured for different types of VN requests under varying offered loads on the network.

4.1. Objectives and performance metrics

We set the penalty functions for CPU and link utilization as follows.

CPU penalty function: Let us denote the CPU utilization by $\beta \in [0, 1]$. We have that

$$\Gamma_{\mathrm{C}}(\beta) = \begin{cases} 12\beta, & 0\% \le \beta < 50\%, \\ 40\beta - 14, & 50\% \le \beta \le 100\%. \end{cases} \tag{23}$$

Link penalty function: For the link penalty function, we adopt the congestion cost function proposed in [41]. Let us denote the link utilization by $\beta \in [0, 1]$. We have that

$$\Gamma_{\mathrm{L}}(\beta) = \begin{cases} \beta, & 0\% \le \beta < 33\%, \\ 3\beta - 2/3, & 33\% \le \beta < 66\%, \\ 10\beta - 16/3, & 66\% \le \beta < 90\%, \\ 70\beta - 178/3, & 90\% \le \beta < 100\%. \end{cases}$$

Performance metrics: Let us denote the revenue earned by the system per unit time at time t by $\mathrm{Rev}(t)$. From (1), we have that

$$\mathrm{Rev}(t) = \sum_{u \in N^S} S_N^{(t)}(v) + \sum_{e \in N^S} S_L^{(t)}(e) \tag{24}$$

where $S_N^{(t)}(v)$ (resp. $S_L^{(t)}(e)$) denote the amount of CPU resource (resp. bandwidth) in use at substrate node v (resp. link e) at time t. We define the time average of the revenue as

$$\overline{\mathrm{Rev}} \ := \lim_{T \to \infty} \frac{1}{T} \int_0^T \mathrm{Rev}(t)dt.$$

We define the power cost at time t by

$$\mathrm{Cost}(t) = \sum_{u \in N^S} P_{\mathrm{SS}}(S_N^{(t)}(u))$$

under the speed scaling model. A similar cost function can be defined for the power-down model. We define the time average of cost as

$$\overline{\mathrm{Cost}} \ := \lim_{T \to \infty} \frac{1}{T} \int_0^T \mathrm{Cost}(t)dt.$$

We define the profit as revenue minus cost. The profit earned by the system per unit time at time t is given by,

$$\mathrm{Pro}(t) \ := \mathrm{Rev}(t) - \mathrm{Cost}(t), \tag{25}$$

and the average profit is defined as

$$\overline{\mathrm{Pro}} \ := \overline{\mathrm{Rev}} - \overline{\mathrm{Cost}}. \tag{26}$$

4.2. Simulation settings

We used the MATLAB tool to implement the proposed algorithms and to perform simulation of arrivals and service of virtual networks. The substrate network has 100 substrate nodes forming 10 × 10 grid topology. The CPU capacity of substrate nodes is given by 400 units. The bandwidth capacity of substrate links is given by 400 units. We consider two types of VNs: fixed and random graphs. The VN with a fixed graph consists of 8 nodes and 7 links. For the VNs with random graphs, we consider Erdős-Renyi graphs where each pair of nodes is connected with probability 0.5, and the number of nodes is randomly selected between 2 and 10.

In our simulations, the VN requests arrive according to a homogeneous Poisson process, and the lifetime of a VN request has the exponential distribution. The coefficient of power consumption ρ_P in the objective is chosen based on the electricity usage of data centers in the U.S. The total electricity usage of data centers in the U.S. is 76.4 billion kWh/y, and the number of servers is 12.2 million [52]. The industrial price of electricity is 6.92 cents/kWh on average [53]. Thus, the electricity fee is about $1.19 per day to which we set ρ_P. The parameters used in our simulation is summarized in Table 2.

Input: substrate nodes $\bar{s}_i$ and $\bar{t}_i$ such that virtual source and sink nodes s_i and t_i for ith virtual link are mapped onto $\bar{s}_i$ and $\bar{t}_i$ respectively in the node mapping phase.
`/* solve MFP */`
Solve **LM**, get $f_e^i, \forall e \in E^S$
`/* randomized rounding */`
`/* path stripping */`
Let $E^{S'} \subseteq E^S$ the set of links such that $f_e^i > 0$. Construct induced graph $G^{S'} = (N^{S'}, E^{S'})$.
$k \leftarrow 0$, $\Lambda \leftarrow \emptyset$ where Λ is a set of paths
while there exists a link $(\bar{s}_i, v) \in E^{S'}$ such that $f^i_{\bar{s}_i,v} > 0$ **do**
  $k \leftarrow k + 1$
  Find the shortest path from $\bar{s}_i$ to $\bar{t}_i$. Denote the path by $P_k := \{e_1, \cdots, e_p\}$
  Add path P_k to Λ
  $\phi_k \leftarrow \min_{j=1,\ldots,p}\{f^i_{e_j}\}$
  $f^i_{e_j} \leftarrow f^i_{e_j} - \phi_k, \forall j = 1, \ldots, p$
  for all $\forall e \in E^{S'}$ **do**
    if $f_e^i = 0$ **then**
      Remove e from $G^{S'}$
    end if
  end for
end while
`/* randomized path selection */`
Assign path P_j in Λ with probability $\phi_j/(\sum_{l=1}^{k} \phi_l)$, $j = 1, \ldots, k$. Select a path from Λ at random according to the probability distribution P_j. Denote the selected path by P^*.
if $\mathrm{BW}(e_i^V) \leq R_L(P^*)$ **then**
  Map the virtual request to P^*
else
  Reject the request G^V
end if

Algorithm 3. Link mapping for ith virtual link.

Table 2
Parameters in simulation.

Parameter	Value
Substrate node capacity	400
Substrate link bandwidth	400
Duration of lifetime	100, 200
Mean arrival rate	0.1, 0.15, 0.2, 0.25, 0.3
μ, α (SS)	0.001, 2
μ, σ, τ, δ (PD)	0.03, 5, 13, 0.99
ρ_P, ρ_C	1.19, 1

4.3. Simulation results

We evaluate the performance of our scheme separately for different energy models, using performance metrics such as revenue, power cost, acceptance ratio and profit. We compare our scheme with the well-known ones such as D-VINE, R-VINE [13] and TR-CL [15].

4.3.1. Revenue and profit under speed scaling

We consider VNs with the fixed graph. The CPU requirement of each VN node is randomly selected between 120 and 140 units, and the bandwidth requirement of each VN link is randomly selected between 20 and 30 units. The mean arrival rate of VN requests equal to 0.2. The average lifetime of a VN request is 100 time units. Fig. 7(a)–(c) show the revenue, acceptance ratio and profit averaged over a time window of interest. For example, the revenue plotted at window value W is Rev (t) averaged over time interval [0, W]. We observe that the average revenue, acceptance ratio and profit of the proposed scheme are higher than other algorithms. Fig. 7(d)–(f) show the time averages of revenue, power cost, acceptance ratio and profit against the arrival rates of VN requests. The plot shows long-term averages, i.e., $\overline{\text{Rev}}$, the average acceptance ratio, and $\overline{\text{Pro}}$. We observe that the proposed scheme outperforms the other schemes over all the simulated arrival rates. Our scheme obtains revenue about 19%–41% and 27%–70% higher than D-VINE and R-VINE. As a result, our scheme earns 22%–40% and 30%–64% higher profit as compared to D-VINE and R-VINE.

We observe that the performance gain of our scheme relative to others increases with the arrival rate. This is because, as the arrival rate increases and the resources get congested, the load balancing aspect becomes more important for performance optimization. Indeed our scheme focuses on load balancing, which is also advantageous to power saving under the speed scaling model. Thus the proposed scheme is more efficient not only in utilizing resources but also in spending power, which results in higher gains in revenue and power saving, leading to larger gaps in profit gains.

4.3.2. Power costs under speed scaling

Although we did not show the power costs in the previous simulation, our scheme incurred the highest power costs. However, this is because the acceptance ratio under our scheme is the highest, and thus our scheme served the highest number of VN requests. Thus, in order for a fair comparison of power saving performance, we must make the offered load to the system, i.e., the product of acceptance ratio and arrival rate, equal for different schemes, as follows.

Fig. 8(a) shows the power costs averaged over time windows, and Fig. 8(b) shows the plot of $\overline{\text{Cost}}$ against arrival rates. We consider VNs with random graphs. The CPU requirement of a VN node is randomly selected between 100 and 120, the bandwidth request of VN link is randomly selected from 10 to 20, and the mean lifetime of a VN request is 100 units. The acceptance ratio of all the cases is 100%. Thus, the offered load to the system is identical for all the schemes. We observe that, the proposed algorithm saves more power than other schemes. We reduce the power consumption by 13%–26% and 14%–27% compared to D-VINE and R-VINE respectively.

By combining the overall results on revenue, profit and power saving, we conclude that our scheme outperforms others both in

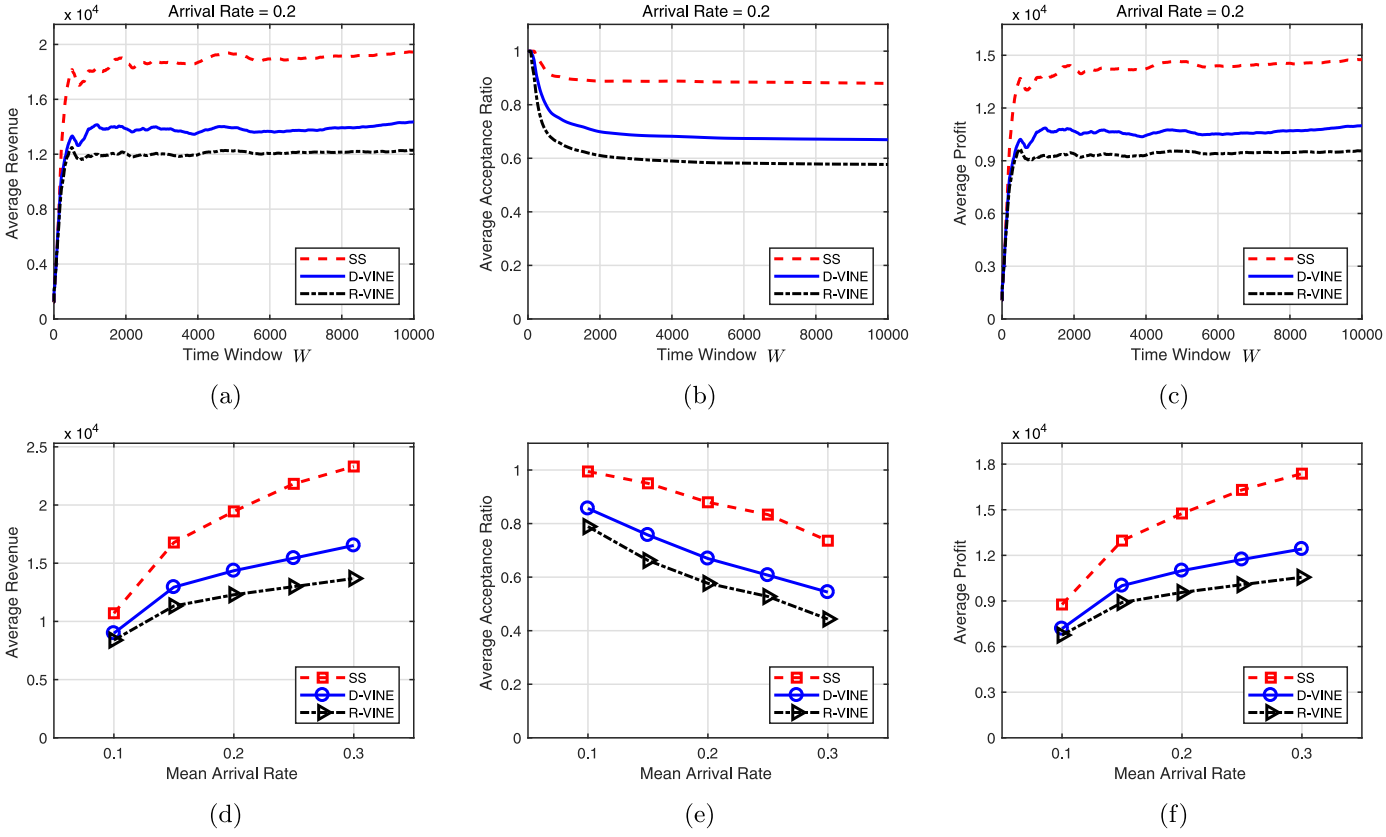

Fig. 7. Average revenue, acceptance ratio and profit under speed scaling.

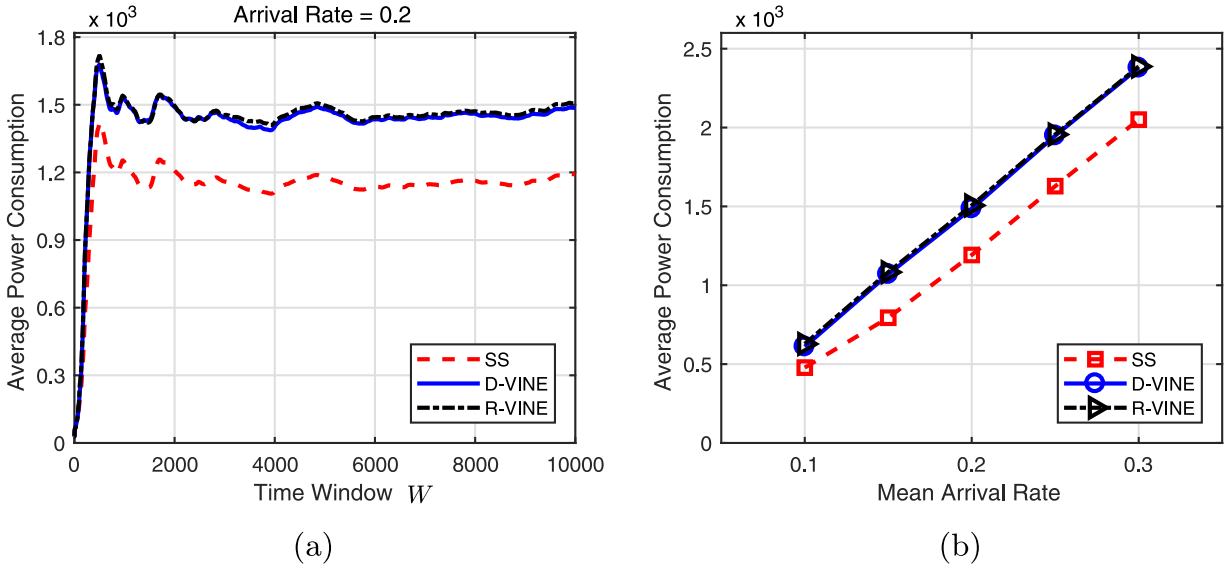

Fig. 8. Average power consumption under speed scaling.

revenue generation by load balancing and cost reduction by power saving, under the speed scaling model.

4.3.3. Revenue and profit under power-down

We evaluate the performance of our algorithm under the power-down model. To benchmark the proposed algorithm, we consider the TR-CL algorithm recently proposed in [15]. TR-CL is an energy-efficient heuristic for the power-down model which works well for moderate to large size networks, in contrast to other algorithms [34,35] based on mixed integer programming. Also we include D-VINE and R-VINE in our comparison. We show that the proposed algorithm can achieve various trade-off points between load balancing and power saving. The degree of trade-offs can be controlled by parameter $\theta \in [0, 1]$ in Algorithm 2, whereas $\theta = 0$ represents maximal power saving, whereas $\theta = 1$ represent maximal load balancing.

Fig. 9(a)–(c) show the average revenue, acceptance ratio and profit over time window $[0, W]$. We consider VNs with the fixed graph. The CPU requirement of a VN node is randomly selected between 130 and 140 units, the bandwidth requirement of a VN link is randomly selected from 10 to 20 units, the mean lifetime of a VN request is 200 units, and the mean arrival rate of VN requests is 0.2. We observe that, the acceptance ratio improves as θ increases. This is because, larger θ means that the algorithm puts more emphasis on load balancing, which result in higher acceptance ratio and revenue. The proposed algorithm with $\theta = 1$ performs the best in terms of acceptance ratio, revenue and profit. We observe that, if our algorithm is operated for maximal power saving, or $\theta = 0$, it still achieves better acceptance ratio compared to TR-CL.

Fig. 9(d)–(f) show $\overline{\text{Rev}}$, $\overline{\text{Cost}}$ and $\overline{\text{Pro}}$ against varying arrival rates of VN requests. We observe that, as parameter θ increases, the acceptance ratio, revenue and profit changes improve. The profit earned by the system under our scheme with $\theta = 1$ is higher than TR-CL, D-VINE and R-VINE, by 91%–134%, 17%–20% and 46%–56% respectively. Note that TR-CL focuses only on power saving by design. However, our scheme with $\theta = 0$, i.e., with maximal power saving, outperforms TR-CL, achieving higher acceptance ratio and profit. This is because our link mapping considers load balancing by reducing link congestions, whereas TR-CL simply uses shortest path routing. Such load balancing aspects of our algorithm lead to better resource utilization and thus improved acceptance ratio.

4.3.4. Power costs under power-down

Next, we examine the performance of power saving of the proposed algorithm. Fig. 10(a) and (b) show the power costs of the schemes in comparison. The experimental setup for VNs is identical to that for simulating power costs under speed scaling. The acceptance ratio is 100%, hence the offered loads to the system is identical across the schemes. We observe that, the power consumption reduces with decreasing θ, as is expected from our algorithm design. TR-CL indeed performs the best; however, our scheme with $\theta = 0$, or with maximal power saving, performs nearly identical to TR-CL. Our scheme not only achieves a lower power consumption than D-VINE for $\theta = 0$, 0.25 and 0.5, but also achieves a lower power consumption than R-VINE for $\theta = 0$ and 0.25.

In summary, we observe that, our algorithm with $\theta = 0.5$ and 0.25 overall performs close to D-VINE and R-VINE, respectively, in terms of revenue, profit and power saving. However, depending on relative importance between revenue and power, our algorithm can be tuned to yield either higher revenue or lower power consumption. In terms of power saving, our algorithm with $\theta = 0$ performs similar to TR-CL, showing that our scheme has a good capability of power saving. Moreover, our algorithm outperforms TR-CL with higher offered loads to the system, thanks to the load balancing effect of the link mapping algorithm. Our simulation demonstrates that our scheme is able to achieve a considerably wide range of revenue-power trade-offs for power-down model, by a simple adjustment of parameter θ. This clearly shows that our algorithm is versatile, i.e., it can be adapted to scenarios requiring varying degrees of revenue-power trade-offs.

5. Conclusion

In this paper, we proposed a VN embedding algorithm which jointly captures load balancing and energy savings to maximize the profit earned by infrastructure providers. In order to maximize profit, we attempted to minimize the penalty function for revenue maximization and load balancing, combined with the power cost function. For power saving we considered both speed scaling and power-down models. We showed that, in speed scaling model, the load balancing and power saving have the objectives with similar properties, and thus allows an efficient joint optimization. However, they become conflicting objectives in power-down model, and we proposed iterative node mapping

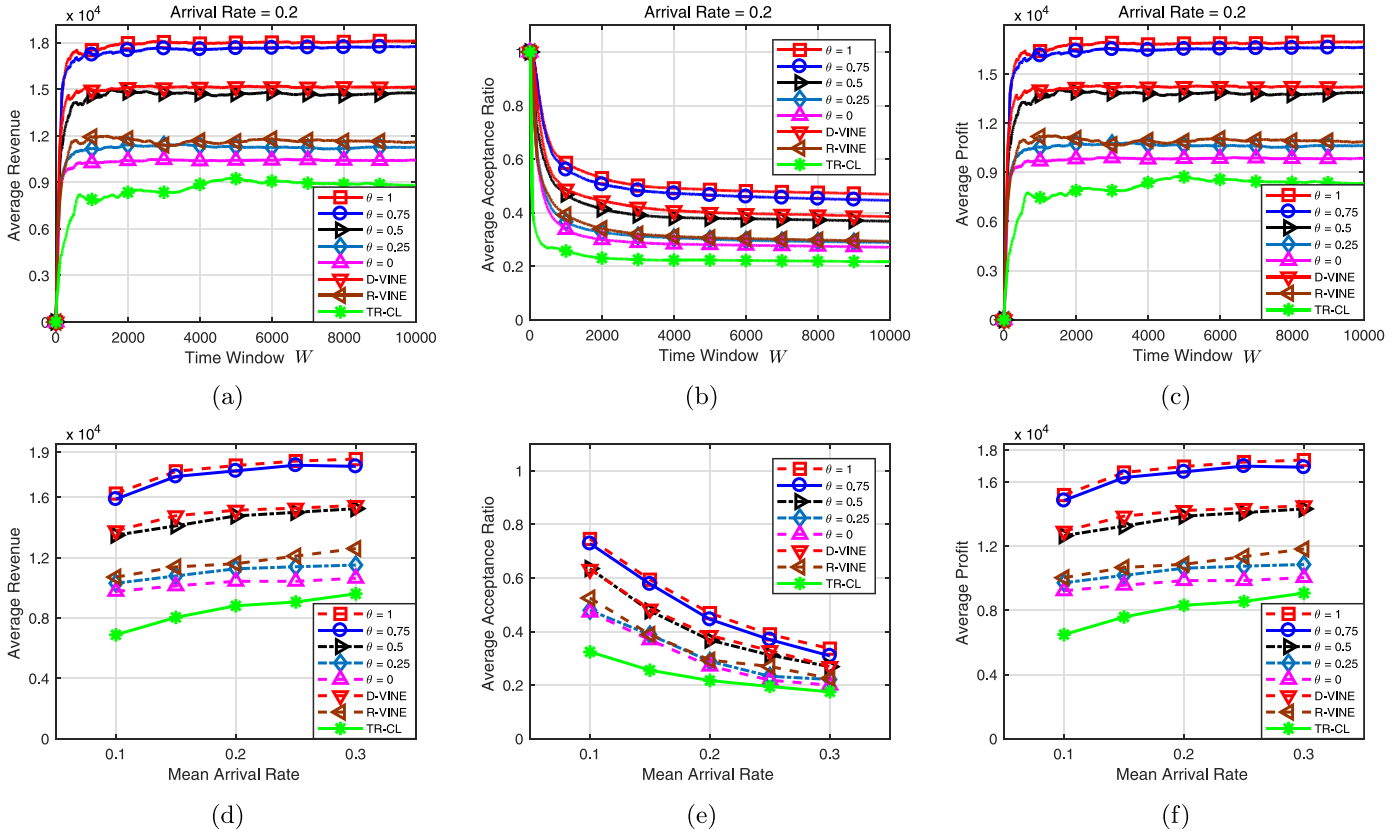

Fig. 9. Average revenue, acceptance ratio and profit under power-down.

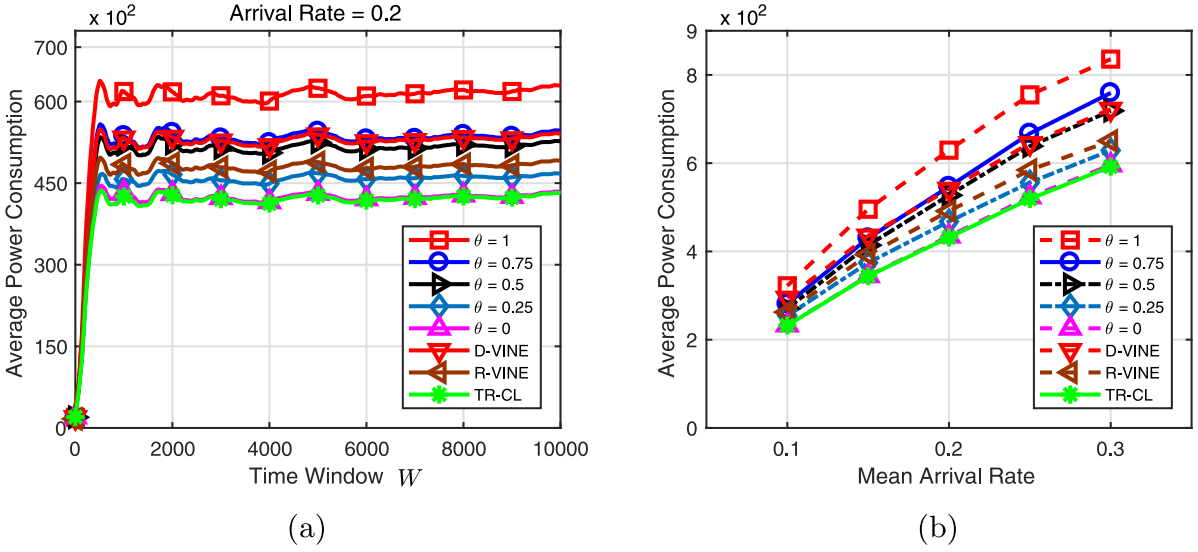

Fig. 10. Average power consumption under power-down.

with re-balancing which explores various balance points between those objectives. We demonstrated via simulation that, our algorithm is effective in both load balancing and power saving for the speed scaling model, and is able to achieve various trade-off points between acceptance ratio and power consumption for the power-down model. In our future work, we plan to add reconfiguration features, such as the migration of VNs, to our mapping scheme. In such setup we would like to develop a framework for re-embedding the mapped VNs considering the interaction between power efficiency and re-distribution of loads, and their dependency on the power models.

Acknowledgment

This work is supported in part by National Research Foundation of Korea (NRF) grant funded by theKorea government (MSIP) (NRF-2016R1A2B1014934), and by Institute for Information and communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. B0126-18-1046).

Appendix A. Proof of Proposition 1

Let f_e^* denote the sum of optimal fractional flows on substrate link e, and $\widehat{f_e}$ denote the sum of rounded flows on e. We would like to show $\mathbb{E}[\Gamma_L(\widehat{f_e})] \leq \gamma \Gamma_L(f_e^*)$. We consider two cases.

Case 1 : $f_e^* \leq d$. Let the rounded solution $\widehat{f_e}$ belongs to one of the set of ranges $[0, d)$, $[d, 2d)$, $[2d, 4d)$, … , $[2^j d, 2^{j+1}d)$, … . We have that

$$\begin{aligned}
&\mathbb{E}[\Gamma_L(\widehat{f_e})] \\
&\leq \Gamma_L(0)\cdot\mathbb{P}(\widehat{f_e} < d) + \sum_{j\geq 0} \Gamma_L(2^{j+1}d)\cdot\mathbb{P}(\widehat{f_e} \geq 2^j d) && \text{(A.1)}\\
&\leq 0 + \sum_{j\geq 0} a_{N_L} 2^{j+1} d\cdot\mathbb{P}(\widehat{f_e} \geq 2^j d) && \text{(A.2)}\\
&\leq \left\{ \sum_{j\geq 0} a_{N_L} 2^{j+1} d \left(\frac{\exp\left(\frac{2^j d}{f_e^*} - 1\right)}{\left(\frac{2^j d}{f_e^*}\right)^{\frac{2^j d}{f_e^*}}} \right)^{f_e^*} \right\} && \text{(A.3)}\\
&= \left\{ a_{N_L} f_e^* d \sum_{j\geq 0} \frac{2^{j+1}}{f_e^*} \frac{{f_e^*}^{2^j d}}{\exp(f_e^*)} \left(\frac{\exp(1)}{2^j d}\right)^{2^j d} \right\} \\
&\leq \Gamma_L(f_e^*) \left\{ \frac{a_{N_L}}{a_1} \left(\sum_{j\geq 0} 2^{(j+1)-2^j d\left(j-\frac{1}{\ln 2}\right)} \right) \right\} && \text{(A.4)}
\end{aligned}$$

(A.1) follows from the definition of expectations. In (A.2), we use the following inequality

$$\Gamma_L(2^{j+1}d) \leq a_{N_L} 2^{j+1} d.$$

(A.3) follows from the Chernoff bound [54] and $\mathbb{E}[\widehat{f_e}] = f_e^*$. In (A.4), we use the following inequalities

$$\begin{aligned}
a_{N_L} f_e^* &= \frac{a_{N_L}}{a_1} f_e^* a_1 \leq \frac{a_{N_L}}{a_1} \Gamma_L(f_e^*), \\
\frac{{f_e^*}^{2^j d-1}}{\exp(f_e^*)} &\leq \frac{(d)^{2^j d-1}}{\exp(0)} = (d)^{2^j d-1}.
\end{aligned} \qquad \text{(A.5)}$$

The summation in the parentheses of (A.4) converges to a constant. Thus, there exist a constant γ_1 that guarantees $\mathbb{E}[C(\widehat{f_e})] \leq \gamma_1 \Gamma_L(f_e^*)$.

Case 2 : $f_e^* > d$. Suppose rounded solution $\widehat{f_e}$ is in one of the ranges $[0, f_e^* d)$, $[f_e^* d, 2f_e^* d)$, $[2f_e^* d, 4f_e^* d)$, … , $[2^j f_e^* d, 2^{j+1} f_e^* d)$, …. We have a series of inequalities in (A.6)–(A.10).

$$\begin{aligned}
&\mathbb{E}[\Gamma_L(\widehat{f_e})] \leq \Gamma_L(f_e^* d)\cdot\mathbb{P}(\widehat{f_e} \geq 0) + \sum_{j\geq 0} \Gamma_L(2^{j+1} f_e^* d)\cdot\mathbb{P}(\widehat{f_e} \geq 2^j f_e^* d) && \text{(A.6)}\\
&\leq \frac{a_{N_L}}{a_1}\Gamma_L(f_e^*)d + \sum_{j\geq 0} a_{N_L} 2^{j+1} f_e^* d\cdot\mathbb{P}(\widehat{f_e} \geq 2^j f_e^* d) && \text{(A.7)}\\
&\leq \frac{a_{N_L}}{a_1}\Gamma_L(f_e^*)d + \left\{ a_{N_L} f_e^* d \sum_{j\geq 0} 2^{j+1} \left(\frac{\exp(2^j d - 1)}{(2^j d)^{2^j d}} \right)^{f_e^*} \right\} && \text{(A.8)}\\
&\leq \frac{a_{N_L}}{a_1}\Gamma_L(f_e^*)d + \left\{ a_{N_L} f_e^* d \left(\sum_{j\geq 0} 2^{j+1} \left(\frac{\exp(2^j d - 1)}{(2^j d)^{2^j d}} \right)^{d} \right) \right\} && \text{(A.9)}\\
&\leq \Gamma_L(f_e^*) \left\{ \frac{a_{N_L}}{a_1} d \left(1 + \sum_{j\geq 0} 2^{(j+1)-2^j d^2\left(j-\frac{1}{\ln 2}\right)-\frac{d}{\ln 2}\cdot d^{-2^j d^2}} \right) \right\} && \text{(A.10)}
\end{aligned}$$

(A.6) follows from the definition of expectations. In (A.7), we use

$$\begin{aligned}
\Gamma_L(2^{j+1} f_e^* d) &\leq a_{N_L} 2^{j+1} f_e^* d, \\
\Gamma_L(f_e^* d) &\leq a_{N_L} f_e^* d \\
&\leq \frac{a_{N_L}}{a_1}\Gamma_L(f_e^*)d.
\end{aligned}$$

(A.8) follows from the Chernoff bound and $\mathbb{E}[\widehat{f_e}] = f_e^*$. In (A.9), we use

$$\left(\frac{\exp(2^j d-1)}{(2^j d)^{2^j d}}\right)^{f_e^*} \le \left(\frac{\exp(2^j d-1)}{(2^j d)^{2^j d}}\right)^{d}, \tag{A.11}$$

which is explained as follows. Let $t = 2^j d$ and $y(t) = \frac{\exp(t-1)}{t^t}$, then

$$\frac{dy(t)}{dt} = -\exp(t-1)t^{-t}\ln t. \tag{A.12}$$

(A.12) is greater than or equal to 0 for all $t \le 1$ and it is less than 0 for all $t > 1$. Thus, $y(t)$ achieves its maximum at $t = 1$ given by

$$\left.\frac{\exp(t-1)}{t^t}\right|_{t=1} = 1.$$

Thus, $y(t) \le 1$ holds, and (A.11) follows. Inequality (A.10) is due to (A.5) of **Case 1**. The expression in the bracket of (A.10) is a constant which we denote by γ_2. Thus, we have $\mathbb{E}[\Gamma_L(\widehat{f}_e)] \le \gamma_2 \Gamma_L(f_e^*)$.

From the above two cases, we proved $\mathbb{E}[\Gamma_L(\widehat{f}_e)] \le \gamma \Gamma_L(f_e^*)$, where $\gamma = \max\{\gamma_1, \gamma_2\} > 0$ for all link e. Therefore

$$\begin{aligned}\mathbb{E}\left[\sum_e \Gamma_L(\widehat{f}_e)\right] &= \sum_e \mathbb{E}[\Gamma_L(\widehat{f}_e)] \le \gamma \sum_e \Gamma_L(f_e^*) \\ &\le \gamma \Gamma^*\end{aligned} \tag{A.13}$$

where Γ^* is the optimal integral solution. Inequality (A.13) holds, because in flow problems, the optimal fractional solution is a lower bound on the optimal integral solution.

Appendix B. Proof of Proposition 2

We use a similar technique to the proof of [17, Theorem 6]. We first partition the range of bandwidth demands $[D_1, D_2]$ into several groups as follows. Let $h := \left\lfloor \log \frac{D_2}{D_1} \right\rfloor$, $D_1 := \min_i\{d_i\}$, and $D_2 := \max_i\{d_i\}$. The jth group is defined to be $[2^j D_1, 2^{j+1} D_1)$ for $j = 0, \cdots, h$. We raise all the bandwidth demands in group j to $2^{j+1}D_1$. The randomized rounding is applied and fractional solutions are obtained with respect to these raised demands in group j. Let $f_e^*(j)$ denote the obtained optimal fractional flow, $\widehat{f}_e(j)$ denote the optimal flow rounded from the fractional flow $f_e^*(j)$, and $\overline{\Gamma}^*(j)$ denote the optimal integral solution. Also let $\Gamma^*(j)$ denote the optimal integral solution obtained by the original demands in group j, and Γ^* denote the optimal integral solution obtained by the original demands in all groups. We have that

$$\sum_e \mathbb{E}\left[\Gamma_L\left(\sum_{j=0}^h \widehat{f}_e(j)\right)\right] \le \sum_e \mathbb{E}\left[\frac{a_{N_L}}{a_1}\sum_{j=0}^h \Gamma_L(\widehat{f}_e(j))\right] \tag{B.1}$$

$$\le \sum_e \frac{a_{N_L}}{a_1}\sum_{j=0}^h \gamma \Gamma_L(f_e^*(j)) \tag{B.2}$$

$$\le \frac{a_{N_L}}{a_1}\gamma \sum_{j=0}^h 2\frac{a_{N_L}}{a_1}\Gamma^*(j) \tag{B.3}$$

$$\le 2\gamma\left(\frac{a_{N_L}}{a_1}\right)^2 \Gamma^*. \tag{B.4}$$

In the above derivation, (B.1) is due to

$$\begin{aligned}&\Gamma_L\left(\sum_{j=0}^h \widehat{f}_e(j)\right) \\ &= \sum_{p=1}^{l} a_p(M_p - M_{p-1}) + a_{l+1}\left(\sum_{j=0}^h \widehat{f}_e(j) - M_l\right) \\ &\le a_{l+1}\sum_{j=0}^h \widehat{f}_e(j) = \frac{a_{l+1}}{a_1}a_1 \sum_{j=0}^h \widehat{f}_e(j) \\ &\le \frac{a_{l+1}}{a_1}\sum_{j=0}^h \Gamma_L(\widehat{f}_e(j)) \le \frac{a_{N_L}}{a_1}\sum_{j=0}^h \Gamma_L(\widehat{f}_e(j))\end{aligned}$$

where $l = \max\left\{k \middle| M_k \le \sum_{j=0}^h \widehat{f}_e(j)\right\}$; (B.2) follows from Proposition 1; (B.3) follows from $\sum_e \Gamma_L(f_e^*(j)) \le 2\frac{a_{N_L}}{a_1}\Gamma^*(j)$ which can be derived as follows. Let $\widehat{\Gamma}^*(j)$ denote the optimal integral solution obtained when we double the original bandwidth demands in group j. We have that

$$\sum_e \Gamma_L(f_e^*(j)) \le \overline{\Gamma}^*(j) \le \widehat{\Gamma}^*(j) \le 2\frac{a_{N_L}}{a_1}\Gamma^*(j) \tag{B.5}$$

because solution $\overline{\Gamma}^*(j)$ is obtained from $2^{j+1}D_1$-raised demands which are at most twice the original demands, for which the solution is $\widehat{\Gamma}^*(j)$. Then,

by using the property of function Γ_L, we obtain the last inequality of (B.5). Finally, (B.4) is due to the superadditive property of $\Gamma_L(\cdot)$. That is, $\Gamma_L\left(\sum_{j=0}^{h}\widehat{f_e}(j)\right) \geq \sum_{j=0}^{h}\Gamma_L(\widehat{f_e}(j))$. In conclusion, the expected cost is at most $2\gamma(a_{N_L}/a_1)^2$ times the optimal cost.

References

[1] Y. Liang, S. Zhang, Embedding parallelizable virtual networks, Comput. Commun. 102 (2017) 47–57.
[2] R. Mijumbi, J. Serrat, J.L. Gorricho, N. Bouten, F. De Turck, R. Boutaba, Network function virtualization: state-of-the-art and research challenges, IEEE Commun. Surv. Tutorials 18 (1) (2016) 236–262.
[3] D.B. Rawat, S.R. Reddy, Software defined networking architecture, security and energy efficiency: a survey, IEEE Commun. Surv. Tutorials 19 (1) (2017) 325–346.
[4] S. Singh, R.K. Jha, A survey on software defined networking: architecture for next generation network, J. Netw. Syst. Manage. 25 (2) (2017) 321–374.
[5] Amazon web services, available from https://aws.amazon.com/.
[6] Goole cloud platform, available from https://cloud.google.com/.
[7] Microsoft azure, available from https://azure.microsoft.com/.
[8] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, M. Bowman, Planetlab: an overlay testbed for broad-coverage services, ACM SIGCOMM Comput. Commun. Rev. 33 (3) (2003) 3–12.
[9] Global environment for network innovations, web site https://geni.net/.
[10] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, A. Joglekar, An integrated experimental environment for distributed systems and networks, ACM SIGOPS Oper. Syst. Rev. 36 (SI) (2002) 255–270.
[11] Y. Zhu, M. Ammar, Algorithms for assigning substrate network resources to virtual network components, Proc. IEEE INFOCOM, 2 (2006), pp. 1–12.
[12] M. Yu, Y. Yi, J. Rexford, M. Chiang, Rethinking virtual network embedding: substrate support for path splitting and migration, ACM SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 17–29.
[13] M. Chowdhury, M.R. Rahman, R. Boutaba, ViNEYard: virtual network embedding algorithms with coordinated node and link mapping, IEEE/ACM Trans. Netw. (TON) 20 (1) (2012) 206–219.
[14] S. Su, Z. Zhang, A.X. Liu, X. Cheng, Y. Wang, X. Zhao, Energy-aware virtual network embedding, IEEE/ACM Trans. Netw. 22 (5) (2014) 1607–1620.
[15] X. Chen, C. Li, Y. Jiang, Optimization model and algorithm for energy efficient virtual node embedding, IEEE Commun. Lett. 19 (8) (2015) 1327–1330.
[16] A. Shehabi, S. Smith, N. Horner, I. Azevedo, R. Brown, J. Koomey, E. Masanet, D. Sartor, M. Herrlin, W. Lintner, United States data center energy usage report, LBNL-1005775, Lawrence Berkeley National Laboratory, Berkeley, 4, California, 2016.
[17] M. Andrews, A.F. Anta, L. Zhang, W. Zhao, Routing for energy minimization in the speed scaling model, INFOCOM, 2010 Proceedings IEEE, IEEE, 2010, pp. 1–9.
[18] X. Fan, W.D. Weber, L.A. Barroso, Power provisioning for a warehouse-sized computer, ACM SIGARCH Computer Architecture News, volume 35, ACM, 2007, pp. 13–23.
[19] S. Rivoire, P. Ranganathan, C. Kozyrakis, A comparison of high-level full-system power models. HotPower, volume 8, (2008). 3–3
[20] P. Raghavan, C.D. Tompson, Randomized rounding: a technique for provably good algorithms and algorithmic proofs, Combinatorica, volume 7, (1987), pp. 365–374.
[21] Y. Zhou, Y. Li, D. Jin, L. Su, L. Zeng, A virtual network embedding scheme with two-stage node mapping based on physical resource migration, Communication Systems (ICCS), 2010 IEEE International Conference on, IEEE, 2010, pp. 761–766.
[22] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, J. Wang, Virtual network embedding through topology-aware node ranking, ACM SIGCOMM Comput. Commun. Rev. 41 (2) (2011) 38–47.
[23] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervello-Pastor, A. Monje, On the optimal allocation of virtual resources in cloud computing networks, IEEE Trans. Comput. 62 (6) (2013) 1060–1071.
[24] S. Zhang, Z. Qian, J. Wu, S. Lu, L. Epstein, Virtual network embedding with opportunistic resource sharing, IEEE Trans. Parallel Distrib. Syst. 25 (3) (2014) 816–827.
[25] L. Gong, Y. Wen, Z. Zhu, T. Lee, Toward profit-seeking virtual network embedding algorithm via global resource capacity, IEEE INFOCOM 2014-IEEE Conference on Computer Communications, IEEE, 2014, pp. 1–9.
[26] S. Ayoubi, Y. Zhang, C. Assi, A reliable embedding framework for elastic virtualized services in the cloud, IEEE Trans. Netw. Serv. Manage. 13 (3) (2016) 489–503.
[27] M.C. Luizelli, L.R. Bays, L.S. Buriol, M.P. Barcellos, L.P. Gaspary, How physical network topologies affect virtual network embedding quality: a characterization study based on isp and datacenter networks, J. Netw. Comput. Appl. 70 (2016) 1–16.
[28] R. Stanojevic, R. Shorten, Distributed dynamic speed scaling, in: INFOCOM, 2010 Proceedings IEEE, IEEE, 2010, pp. 1–5.
[29] K. Son, B. Krishnamachari, Speedbalance: speed-scaling-aware optimal load balancing for green cellular networks, INFOCOM, 2012 Proceedings IEEE, IEEE, 2012, pp. 2816–2820.
[30] J. Kwak, O. Choi, S. Chong, P. Mohapatra, Processor-network speed scaling for energy–delay tradeoff in smartphone applications, IEEE/ACM Trans. Netw. 24 (3) (2016) 1647–1660.
[31] J. Kwak, Y. Kim, J. Lee, S. Chong, Dream: dynamic resource and task allocation for energy minimization in mobile cloud systems, IEEE J. Sel. Areas Commun. 33 (12) (2015) 2510–2523.
[32] L. Chen, N. Li, On the interaction between load balancing and speed scaling, IEEE J. Sel. Areas Commun. 33 (12) (2015) 2567–2578.
[33] S. Su, Z. Zhang, X. Cheng, Y. Wang, Y. Luo, J. Wang, Energy-aware virtual network embedding through consolidation, Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on, IEEE, 2012, pp. 127–132.
[34] J.F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, H. De Meer, Energy efficient virtual network embedding, IEEE Communications Letters, volume 16, (2012), pp. 756–759.
[35] J.F. Botero, X. Hesselbach, Greener networking in a network virtualization environment, Comput. Netw. 57 (9) (2013) 2021–2039.
[36] L. Nonde, T.E. El-Gorashi, J.M. Elmirghani, Energy efficient virtual network embedding for cloud networks, J. Lightwave Technol. 33 (9) (2015) 1828–1849.
[37] G. Sun, V. Anand, D. Liao, C. Lu, X. Zhang, N.H. Bao, Power-efficient provisioning for online virtual network requests in cloud-based data centers, IEEE Syst. J. 9 (2) (2015) 427–441.
[38] L. Wang, F. Zhang, J.A. Aroca, A.V. Vasilakos, K. Zheng, C. Hou, D. Li, Z. Liu, Greendcn: a general framework for achieving energy efficiency in data center networks, selected areas in communications, IEEE J. 32 (1) (2014) 4–15.
[39] M. Andrews, S. Antonakopoulos, L. Zhang, Minimum-cost network design with (dis) economies of scale, Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on, IEEE, 2010, pp. 585–592.
[40] X. Chen, C. Li, Y. Jiang, A feedback control approach for energy efficient virtual network embedding, Comput. Commun. 80 (2016) 16–32.
[41] B. Fortz, M. Thorup, Internet traffic engineering by optimizing ospf weights, INFOCOM 2000. Nineteenth annual joint conference of the IEEE computer and communications societies. Proceedings. IEEE, volume 2, IEEE, 2000, pp. 519–528.
[42] M. Shafiee, J. Ghaderi, A simple congestion-aware algorithm for load balancing in datacenter networks, IEEE/ACM Trans. Netw. 25 (6) (2017) 3670–3682.
[43] N. Bansal, T. Kimbrel, K. Pruhs, Speed scaling to manage energy and temperature, J. ACM (JACM) 54 (1) (2007) 3.
[44] L. Yuan, G. Qu, Analysis of energy reduction on dynamic voltage scaling-enabled systems, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 24 (12) (2005) 1827–1837.
[45] A. Wierman, L.L. Andrew, A. Tang, Power-aware speed scaling in processor sharing systems, INFOCOM 2009, IEEE, IEEE, 2009, pp. 2007–2015.
[46] X. León, L. Navarro, Limits of energy saving for the allocation of data center resources to networked applications, INFOCOM, 2011 Proceedings IEEE, IEEE, 2011, pp. 216–220.
[47] X. León, L. Navarro, A stackelberg game to derive the limits of energy savings for the allocation of data center resources, Future Gener. Comput. Syst. 29 (1) (2013) 74–83.
[48] G. Lu, C. Guo, Y. Li, Z. Zhou, T. Yuan, H. Wu, Y. Xiong, R. Gao, Y. Zhang, Serverswitch: a programmable and high performance platform for data center networks. in: Nsdi, volume 11, (2011). 2–2
[49] V. Sivaraman, A. Vishwanath, Z. Zhao, C. Russell, Profiling per-packet and per-byte energy consumption in the netfpga gigabit router, Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on, IEEE, 2011, pp. 331–336.
[50] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[51] S. Abdelwahab, B. Hamdaoui, M. Guizani, T. Znati, Efficient virtual network embedding with backtrack avoidance for dynamic wireless networks, IEEE Trans. Wireless Commun. 15 (4) (2016) 2669–2683.
[52] P. Delforge, J. Whitney, Data center efficiency assessment, natural resources defense council (NRDC), 2014, https://www.nrdc.org/energy/files/data-center-efficiency-assessment-IP.pdf.
[53] Electric power monthly with data for november 2015, 2016, https://www.eia.gov/electricity/monthly/pdf/epm.pdf.
[54] M. Mitzenmacher, E. Upfal, Probability and Computing: Randomized Algorithms and Probabilistic Analysis, Cambridge University Press, 2005.