Greenhead: Virtual Data Center Embedding across Distributed Infrastructures

Ahmed Amokrane, *Student Member*, *IEEE*, Mohamed Faten Zhani, *Member*, *IEEE*, Rami Langar, *Member*, *IEEE*, Raouf Boutaba, *Fellow*, *IEEE*, and Guy Pujolle, *Member*, *IEEE*

Abstract—Cloud computing promises to provide on-demand computing, storage, and networking resources. However, most cloud providers simply offer virtual machines (VMs) without bandwidth and delay guarantees, which may hurt the performance of the deployed services. Recently, some proposals suggested remediating such limitation by offering virtual data centers (VDCs) instead of VMs only. However, they have only considered the case where VDCs are embedded within a single data center. In practice, infrastructure providers should have the ability to provision requested VDCs across their distributed infrastructure to achieve multiple goals including revenue maximization, operational costs reduction, energy efficiency, and green IT, or to simply satisfy geographic location constraints of the VDCs. In this paper, we propose Greenhead, a holistic resource management framework for embedding VDCs across geographically distributed data centers connected through a backbone network. The goal of Greenhead is to maximize the cloud provider's revenue while ensuring that the infrastructure is as environment-friendly as possible. To evaluate the effectiveness of our proposal, we conducted extensive simulations of four data centers connected through the NSFNet topology. Results show that Greenhead improves requests' acceptance ratio and revenue by up to 40 percent while ensuring high usage of renewable energy and minimal carbon footprint.

Index Terms—Green computing, energy efficiency, cloud computing, virtual data center, distributed infrastructure

1 Introduction

Cloud computing has recently gained significant popularity as a cost-effective model for hosting large-scale online services in large data centers. In a cloud computing environment, an infrastructure provider (InP) partitions the physical resources inside each data center into virtual resources (e.g., virtual machines (VMs)) and leases them to service providers (SPs) in an on-demand manner. On the other hand, a service provider uses those resources to deploy its service applications, with the goal of serving its customers over the Internet.

Unfortunately, current InPs like Amazon EC2 [1] mainly offer resources in terms of virtual machines without providing any performance guarantees in terms of bandwidth and propagation delay. The lack of such guarantees affects significantly the performance of the deployed services and applications [2]. To address this limitation, recent research proposals urged cloud providers to offer resources to SPs in the form of virtual data centers (VDCs) [3]. A VDC is a collection of virtual machines, switches, and routers that are interconnected through virtual links. Each virtual link is characterized by its bandwidth capacity and its propagation delay. Compared to traditional VM-only offerings, VDCs are able to provide better isolation of network resources, and thereby improve the performance of service applications.

Despite its benefits, offering VDCs as a service introduces a new challenge for cloud providers called the VDC embedding problem, which aims at mapping virtual resources (e.g., virtual machines, switches, routers) onto the physical infrastructure. So far, few works have addressed this problem [2], [4], [5], but they only considered the case where all the VDC components are allocated within the same data center. Distributed embedding of VDCs is particularly appealing for SPs as well as InPs. In particular, an SP uses its VDC to deploy various services that operate together to respond to end-users requests. As shown in Fig. 1, some services may require to be in the proximity of end users (e.g., web servers), whereas others may not have such location constraints and can be placed in any data center (e.g., MapReduce jobs).

On the other hand, InPs can also benefit from embedding VDCs across their distributed infrastructure. In particular, they can take advantage of the abundant resources available in their data centers and achieve various objectives including maximizing revenue, reducing costs and improving the infrastructure sustainability.

In this paper, we propose a management framework able to orchestrate VDC allocation across a distributed cloud infrastructure. The main objectives of such framework can be summarized as follows:

- *Maximize revenue*. Certainly, the ultimate objective of an infrastructure provider is to increase its revenue by maximizing the amount of leased resources and the number of embedded VDC requests. However, embedding VDCs requires satisfying different constraints, namely, the capacity and location constraints. Obviously, the embedding scheme must

- *A. Amokrane, R. Langar, and G. Pujolle are with the LIP6/UPMC—University of Pierre and Marie Curie, 4 Place Jussieu, Paris 75005, France.*
  E-mail: {ahmed.amokrane, rami.langar, guy.pujolle}@lip6.fr.
- *M.F. Zhani and R. Boutaba are with the David R. Cheriton School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, Ontario N2L 3G1, Canada.*
  E-mail: {mfzhani, rboutaba}@uwaterloo.ca.

Manuscript received 21 Mar. 2013; revised 6 Aug. 2013; accepted 3 Sept. 2013; published online 12 Sept. 2013.
Recommended for acceptance by I. Stojmenovic.
For information on obtaining reprints of this article, please send e-mail to: tcc@computer.org, and reference IEEECS Log Number TCC-2013-03-0051.
Digital Object Identifier no. 10.1109/TCC.2013.5.

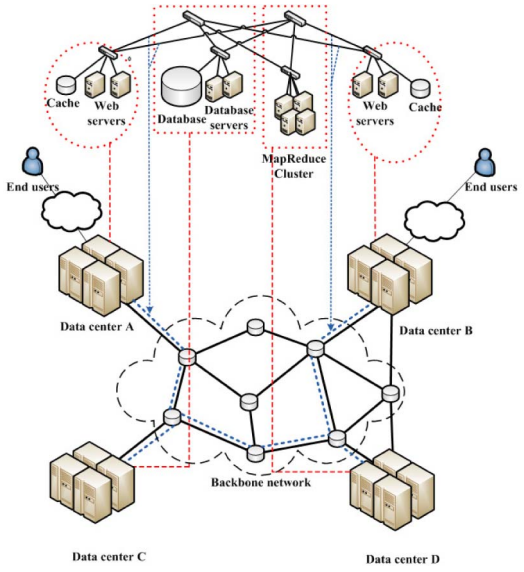


Fig. 1. Example of VDC deployment over a distributed infrastructure.

ensure that the capacity of the infrastructure is never exceeded. In addition, it must satisfy location constraints imposed by SPs.

- *Reduce backbone network workload*. To cope with the growing traffic demand between data centers, infrastructure providers tend to build their proprietary wide-area backbone network to interconnect their facilities (e.g., Google G-scale network [6]). In this context, one key objective when embedding VDCs is to minimize the traffic within the backbone network. Indeed, it has been reported recently that the cost of building an inter-data center network is much higher than the intra-data center network cost and it accounts for 15 percent of the total infrastructure cost [7]. In addition, according to several studies [8], wide-area data transport is bound to be the major contributor to the data transport costs. Hence, it is crucial to reduce the backbone network traffic and place high-communicating VMs within the same data center whenever possible.
- *Reduce data center operational costs*. Reducing data centers' operational costs is a critical objective of any infrastructure provider as it impacts its budget and growth. This can be achieved through minimizing energy costs, which constitutes a significant portion of the total operational expenditure. To this end, two key techniques can be adopted: 1) placing more workload into the most energy-efficient data centers, and 2) taking advantage of the difference in electricity prices between the locations of the infrastructure facilities. In particular, energy-efficient data centers can be identified by their power usage effectiveness (PUE), and favored to host more virtual machines.

  Furthermore, InPs can achieve more savings by considering the fluctuation of electricity price over time and the price difference between the locations of the data centers. Hence, VMs can be efficiently placed such that the total electricity cost is minimized.
- *Reduce the carbon footprint*. Recent research has reported that, in 2012, the carbon footprint of data centers around the world accounted for 0.25 percent of the worldwide carbon emission, which constitutes 10 percent of information and communication technologies (ICT) emissions [9]. As a result, InPs are facing a lot of pressure to operate on renewable sources of energy (e.g., solar and wind power) to make their infrastructure more green and environment-friendly. Based on these observations, an efficient VDC embedding scheme should maximize the usage of renewables and take into account their availability, which depends on the data center geographical location, the time of the day (e.g., day and night for solar power) as well as the weather conditions (e.g., wind, atmospheric pressure). Furthermore, whenever the power from the electric grid is used, the VDC embedding scheme has to minimize the infrastructure carbon footprint. In that case, the placement of the VMs is critical since the carbon footprint per watt of power varies from location to location.

In this paper, we propose Greenhead, a resource management framework for VDC embedding across a distributed infrastructure. Greenhead aims at maximizing the InP's revenue by minimizing energy costs, while ensuring that the infrastructure is as environment-friendly as possible. To reduce the complexity of the problem, we propose a two-step approach. We first divide a VDC request into partitions such that the interpartition bandwidth demand is minimized and the intrapartition bandwidth is maximized. The aim of such partitioning is to embed VMs exchanging high volumes of data in the same data center. This significantly reduces the traffic carried by the backbone, and thereby improves requests' acceptance ratio. We then propose a simple yet efficient algorithm for assigning partitions to data centers based on electricity prices, data centers' PUEs, availability of renewables, and the carbon footprint per unit of power.

To the best of our knowledge, this is the first effort to address VDC embedding problem over a distributed infrastructure taking into account energy efficiency as well as environmental considerations.

The remainder of this paper is organized as follows: In Section 2, we present related works relevant to ours. We then describe the proposed management framework in Section 3. We provide a mathematical formulation of the VDC embedding problem across a distributed infrastructure in Section 4. Section 5 presents a detailed description of the proposed algorithms for VDC partitioning and embedding. Section 6 discusses the simulation results showing the effectiveness of Greenhead. We finally provide concluding remarks in Section 7.

2 Related Work

In this section, we survey relevant research in the literature. We classified previous work into three related topics, namely: VDC embedding within a single data center, virtual network embedding, and workload scheduling across geographically distributed data centers.

2.1 VDC Embedding within a Single Data Center

So far, only few works have addressed the VDC embedding problem. For instance, Guo et al. [5] proposed a data center network virtualization architecture called SecondNet that

incorporates a greedy algorithm to allocate resources to VDCs. Ballani et al. [2] proposed two abstractions for VDCs, namely a virtual cluster and an oversubscribed virtual cluster. They developed Oktopus, an implementation of those abstractions that uses a greedy algorithm for mapping virtual resources to a tree-like physical topology. Finally, Zhani et al. [4] presented VDC Planner, a resource management framework for data centers that leverages dynamic VM migration to improve the acceptance ratio of VDCs, and thereby increases InP's revenue. They also proposed a VDC consolidation algorithm to minimize the number of active physical servers during low-demand periods. Unfortunately, the above proposals cannot be directly applied to allocate resources in multiple data centers due to the large size of the resulting topology. In addition, for a distributed environment, different considerations should be taken into account such as carbon footprint of the data centers and variability of electricity prices over time and between different locations.

The only work we are aware of that addressed multidata center embedding problem is that of Xin et al. [10]. They proposed an algorithm that uses *minimum k-cut* to split a request into partitions before assigning them to different locations. However, this work has only aimed at load balancing the workload through request partitioning without considering other objectives like revenue maximization, backbone network usage optimization, energy efficiency, and green IT. Furthermore, it does not consider constraints on the VM placement.

2.2 Virtual Network Embedding

Virtual network embedding has been extensively studied in the literature. It basically aims at embedding virtual nodes (mainly routers) and links on top of a physical backbone substrate. Current proposals have addressed the embedding problem either in a single domain (i.e., a backbone owned and managed by a single InP) or in multiple domains (i.e., multiple networks managed by different InPs).

In the single domain case, the InP tries to embed the virtual networks while aiming to achieve multiple objectives including: 1) minimizing the embedding cost [11], 2) maximizing the acceptance ratio and revenue [12], [13], and 3) improving energy efficiency [14], [15].

In the multidomain case, the request is provisioned across multiple domains belonging to different InPs. Houidi et al. [16] proposed a centralized approach where the SP first splits the request using Max-Flow Min-Cut based on prices offered by different InPs then decides where to place the partitions. Chowdhury et al. [17] proposed a distributed embedding solution called PolyVine. In PolyVine, the virtual network request is sent to a single InP, which tries to allocate as much resources as possible in his own network before forwarding the unembedded nodes and links to a neighboring provider. The process continues recursively until the whole request is embedded.

The above proposals on virtual network embedding cannot be directly applied to the VDC embedding problem for many reasons. While a virtual network can be made of tens of nodes (mainly routers), a VDC, expected to be similar to a real data center, may comprise thousands of nodes of different types (e.g., VMs, virtual switches and routers). There is, therefore, a definite need for developing new solutions able to embed large-scale VDCs and to consider the diversity of resources. Finally, previous works do not take advantage of the variability of electricity prices between different locations and also ignore environmental considerations.

2.3 Workload Placement in Geographically Distributed Data Centers

Several works have addressed the problem of workload placement in geographically distributed data centers. They either aimed at reducing energy costs [18], [19], [20], or minimizing the carbon footprint [21], [22] or both [23].

Generally, energy costs are cut down by taking advantage of the variability of electricity prices between different data centers and even at the same location over time. The carbon footprint is reduced by following the renewables available during some periods of the day. For instance, Zhang et al. [18] used a model predictive control framework for service placement. Services are dynamically placed in data centers and migrated according to the demand and price fluctuation over time while considering the migration cost and the latency between services and end users. Qureshi et al. [20] addressed the problem of replica placement and request routing in content distribution networks (CDN). They aimed at reducing electricity costs by dynamically placing data at locations with low electricity prices. Gao et al. [24] addressed the same problem, but they aimed at minimizing energy costs, carbon footprint, and the delay between end users and the location of the data. Liu et al. proposed a framework for workload assignment and dynamic workload migration between data centers that minimizes the latency between end users and services while following renewables and avoiding using power from the electricity grid [21], [22]. Le et al. [25] proposed a workload assignment framework across multiple data centers that minimizes the costs of energy consumed by IT and cooling equipment depending on the fluctuations of electricity prices and the variability of the data centers' PUEs.

However, the main limitation of the aforementioned proposals is that they ignore the communication patterns and the exchanged data between the VMs. This makes such approaches not applicable for embedding VDCs since we need also to consider bandwidth and delay requirements between the VDC components.

In summary, our work is different from traditional virtual network and VDC embedding proposals since it considers resource allocation for VDCs across the whole infrastructure including data centers and the backbone network connecting them. It also differs from service placement works since we are provisioning all types of resources including computing, storage, and notably networking (i.e., bandwidth and delay).

To the best of our knowledge, this work is the first effort to address a VDC embedding across a distributed infrastructure while considering the usage of the backbone network, the variability of electricity prices, energy efficiency as well as environmental considerations.

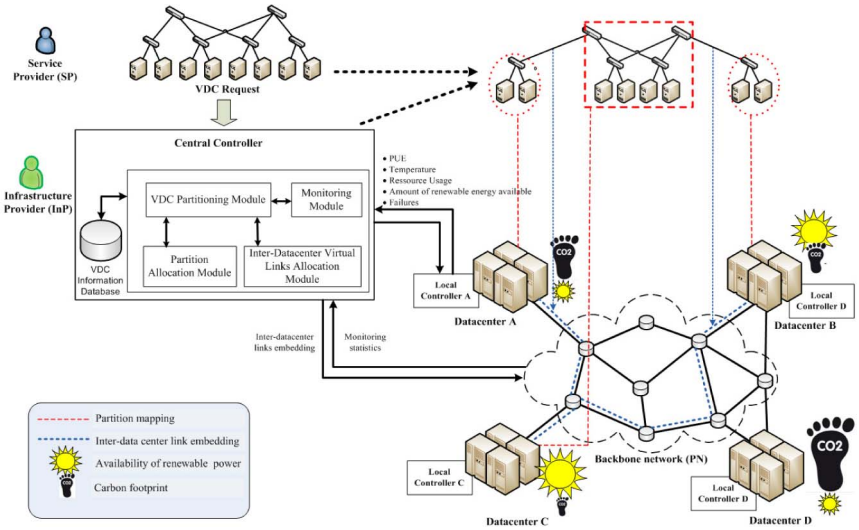


Fig. 2. VDC embedding across multiple data centers.

3 System Architecture

In this work, we consider a distributed infrastructure consisting of multiple data centers located in different regions and interconnected through a backbone network (see Fig. 2). The entire infrastructure (including the backbone network) is assumed to be owned and managed by the same infrastructure provider. Each data center may operate on on-site renewable energy (e.g., wind, solar) and resorts to electricity grid only when its on-site renewable energy becomes insufficient. Unfortunately, renewables are not always available as they depend on the data center location, the time of the day and external weather conditions. While renewable energy has no carbon footprint, energy from the Grid is usually produced by burning coal, oil, and gas, generating high levels of carbon emissions. As a result, whenever electricity is drawn from the Grid, the cloud provider has to pay a penalty proportional to the generated carbon emission. The generated carbon depends on the source of power used by the electric grid supplier, which could be a renewable source or a conventional one or a mix of both. Furthermore, it is also worth noting that prices of the grid electricity differ between regions and they even vary over time in countries with deregulated electricity markets.

As shown in Fig. 2, an SP sends the VDC request specifications to the InP, which has the responsibility of allocating the required resources. Naturally, the cloud provider will make use of its distributed infrastructure with the objective of maximizing its revenue and minimizing energy costs and carbon footprint; this is where our proposed management framework, Greenhead, comes into play. Greenhead is composed of two types of management entities: 1) a central controller that manages the entire infrastructure and 2) a local controller deployed at each data center to manage the data center's internal resources.

The central management entity includes five components as depicted in Fig. 2:

- *The Partitioning Module* is responsible for splitting a VDC request into partitions such that interpartition bandwidth is minimized. The aim of this module is to reduce the number of virtual links provisioned between data centers. Each partition is supposed to be entirely embedded into a single data center. The motivation behind such partitioning will be explained in Section 5.1.
- *The Partition Allocation Module* is responsible for assigning partitions to data centers based on runtime statistics collected by the monitoring module. It ensures that all partitions are embedded while achieving cost effectiveness, energy efficiency, and green IT objectives such as reducing energy costs from the power grid and maximizing the use of renewable sources of energy.
- *The Inter-Data Center Virtual Link Allocation Module* allocates virtual links in the backbone network. Those virtual links connect VMs that have been assigned to different data centers.
- *The Monitoring Module* is responsible for gathering different statistics from the data centers. The collected information includes PUE, resource utilization, outdoor temperature, electricity price, and the amount of available renewable energy.
- *The VDC Information Base* contains all information about the embedded VDCs including their partitions and mapping either onto the data centers or the backbone network.

Regarding the local controller at each data center, its main role is to manage the resources within the data center itself. Specifically, it allocates resources for a partition of a VDC as requested by the central controller. If the embedding is not possible (for example, due to unavailability of resources), the local controller notifies the central controller. Subsequently, the partition allocation module will attempt to find another data center able to embed the rejected partition. It is worth noting that different resource allocation schemes can be deployed locally at the data centers (e.g., VDC planner [4], SecondNet [5], Oktopus [2]). Finally, each local controller has to report periodically statistics including PUE, temperature, resource usage, and availability of renewables to the central controller.

TABLE 1
Table of Notations

Notation	Meaning
PUE^i	PUE of data center i
ζ^i	Electricity price in data center i
η^i	on-site renewable power cost in data center i
N^i	Residual renewable power in data center i
C^i	Carbon footprint per unit of power from the power grid in data center i
α_i	Cost per ton of carbon in data center i
z_{ik}^j	A boolean variable indicating whether data center i satisfies the location constraint of VM k of VDC j
x_{ik}^j	A boolean variable indicating whether VM k is assigned to data center i
$f_{e,e'}$	a boolean variable indicating whether the physical link $e \in E$ is used to embed the virtual link $e' \in E^j$
$\mathcal{D}_i^j$	Cost of embedding the VDC request j in data center i
$P_{i,IT}$	Amount of power consumed only by IT equipment (i.e., servers and switches) in data center i
P_i^j	Total power consumed in data center i
σ^r	Price per unit of resource type r
σ^b	Price per unit of bandwidth
c_p	Cost per unit of bandwidth in the backbone network

4 Problem Formulation

In this section, we formally define the VDC embedding problem across multiple data centers as an integer linear program (ILP). Table 1 describes the notations used in our ILP model.

We assume that time is divided into slots $[1, \ldots, T]$. The metrics characterizing each data center (e.g., PUE, electricity price) are measured at the beginning of each time slot and are considered constant during the corresponding time slot. Thus, for readability, we omit the time reference in all variables defined in the remainder of this section.

The physical infrastructure is represented by a graph $G(V \cup W, E)$, where V denotes the set of data centers and W the set of nodes of the backbone network. The set of edges E represents the physical links in the backbone network. Each link is characterized by its bandwidth capacity and propagation delay.

A VDC request j is represented by a graph $G^j(V^j, E^j)$. Each vertex $v \in V^j$ corresponds to a VM, characterized by its CPU, memory, and disk requirements. Each edge $e \in E^j$ is a virtual link that connects a pair of VMs. It is characterized by its bandwidth demand $bw(e)$ and propagation delay $d(e)$. Furthermore, each VDC j has a lifetime T_j. We assume the revenue generated by VDC j, denoted by $\mathcal{R}^j$, to be proportional to the amount of CPU, memory, and bandwidth required by its VMs and links. Let R denote the different types of resources offered by each node (i.e., CPU, memory and disk). The revenue generated by VDC j can be written as follows:

$$\mathcal{R}^j = \left(\sum_{v \in V^j} \sum_{r \in R} C_j^r(v) \times \sigma^r + \sum_{e' \in E^j} bw(e') \times \sigma^b \right), \quad (1)$$

where $C_j^r(v)$ is the capacity of VM v belonging to the VDC j in terms of resource r, and σ^r and σ^b are the selling prices of a unit of resource type r and a unit of bandwidth, respectively.

Furthermore, we assume that each VM $v \in V^j$ may have a location constraint. Therefore, it can only be embedded in a particular set of data centers. To model this constraint, we define

$$z_{ik}^j = \begin{cases} 1 & \text{If the VM } k \text{ of the VDC } j \text{ can be,} \\ & \text{embedded in data center } i, \\ 0 & \text{Otherwise.} \end{cases}$$

as a binary variable that indicates whether a VM k of to VDC j can be embedded in a data center i.

The problem of embedding a given VDC j across the infrastructure involves two steps:

- First, assign each VM $k \in V^j$ to a data center. Hence, we define the decision variable x_{ik}^j as

$$x_{ik}^j = \begin{cases} 1 & \text{If the VM } k \text{ of the VDC } j \text{ is} \\ & \text{assigned to data center } i \\ 0 & \text{Otherwise.} \end{cases}$$

- Second, embed every virtual link belonging to E^j either in the backbone network if it connects two VMs assigned to different data centers or within the same data center, otherwise. To do so, we define the virtual link allocation variable $f_{e,e'}$ as

$$f_{e,e'} = \begin{cases} 1 & \text{If the physical link } e \in E \text{ is used to} \\ & \text{embed the virtual link } e' \in E^j \\ 0 & \text{Otherwise.} \end{cases}$$

Finally, the ultimate objective of the InP when embedding a VDC request is to maximize its profit defined as the difference between the revenue (denoted by $\mathcal{R}^j$) and the total embedding cost, which consists of the embedding cost in the data centers (denoted by $\mathcal{D}^j$) plus the embedding cost in the backbone network $\mathcal{P}^j$. Hence, our problem can be formulated as an ILP with the following objective function:

$$Maximize \quad \mathcal{R}^j - (\mathcal{D}^j + \mathcal{P}^j) \quad (2)$$

Subject to the following constraints (3)-(8):

- A VM has to be assigned to a data center that satisfies its location constraints:

$$x_{ik}^j \leq z_{ik}^j, \ \forall k \in V^j, \forall i \in V. \quad (3)$$

- A VM is assigned to one and only one data center:

$$\sum_{i \in V} x_{ik}^j = 1, \ \forall k \in V^j. \quad (4)$$

- The capacity constraint of the backbone network links should not be exceeded:

$$\sum_{e' \in E^j} f_{e,e'} \times bw(e') \leq sbw(e), \ \forall e \in E, \quad (5)$$

where $sbw(e)$ is the residual bandwidth of the backbone network link e.

- The required propagation delay for every virtual link allocated in the backbone should be satisfied:

$$\sum_{e \in E} f_{e,e'} \times d(e) \leq d(e'),\ \forall e' \in E^j. \tag{6}$$

- The flow conservation constraint given by

$$\begin{aligned} f_{e_1,e'} - f_{e_2,e'} = x_{d(e_1)d(e')} - x_{s(e_2)s(e'),} \forall e_1, e_2 \in E, \\ d(e_1) = s(e_2),\ \forall\ e' \in V^j, \end{aligned} \tag{7}$$

  where $s(e)$ and $d(e)$ denote the source and destination of link e, respectively.
- Furthermore, the central controller should also ensure that each data center is able to accommodate VMs and virtual links assigned to it. To model this constraint, let $G_i^j(V_i^j, E_i^j)$ denote a partition from G^j, where V_i^j and E_i^j are the set of VMs and virtual links belonging to VDC j and assigned to data center i. They can be written as

$$V_i^j = \left\{k \in V^j \mid x_{ik}^j = 1\right\},$$

$$E_i^j = \left\{e' \in E^j \mid s(e') \in V_i^j \text{ and } d(e') \in V_i^j\right\}.$$

  We define the function

$$Embed_i(G_i^j) = \begin{cases} 1 & \text{If data center } i \text{ can} \\ & \text{accommodate } V_i^j \text{ and } E_i^j \\ 0 & \text{Otherwise.} \end{cases}$$

  Hence, to ensure that the data center i can host the assigned VMs and links, we should satisfy:

$$x_{ik}^j \leq Embed_i\left(G_i^j\right),\ \forall k \in V^j,\ \forall i \in V. \tag{8}$$

Let us now focus on the expression of the embedding costs $\mathcal{D}^j$ and $\mathcal{P}^j$ in the data centers and the backbone network, respectively. Recall that these costs are part of the objective function.

The cost of embedding in the data centers. In this work, we evaluate the request embedding cost in the data centers in terms of energy and carbon footprint costs. To do so, we first evaluate the amount of power required to embed the partition G_i^j in a data center i denoted by P_i^j.

Let $P_{i,IT}^j$ denote the amount of power consumed only by IT equipment (i.e., servers and switches) to accommodate G_i^j (expressed in *kilowatt*). This amount of power depends mainly on the local allocation scheme, the current mapping, and the availability of resources at data center i. The power consumed at the data center i by IT equipment and other supporting systems (e.g., cooling) to accommodate the partition G_i^j can be computed as

$$P_i^j = P_{i,IT}^j \times PUE_i, \tag{9}$$

where PUE_i is the power usage effectiveness of data center i. The mix of power used in data center i is given by

$$P_i^j = P_{i,L}^j + P_{i,D}^j, \tag{10}$$

where $P_{i,L}^j$ and $P_{i,D}^j$ denote, respectively, the on-site consumed renewable power and the amount of purchased power from the Grid. Note that the amount of on-site consumed power should not exceed the amount of produced power, which is captured by $P_{i,L}^j \leq RN_i$, where RN_i is the amount of residual renewable power in data center i expressed in kilowatt.

Hence, the embedding cost (expressed in dollar) of the partition G_i^j in data center i can be written as

$$\mathcal{D}_i^j = P_{i,L}^j \times \eta_i + P_{i,D}^j \times (\zeta_i + \alpha_i C_i), \tag{11}$$

where η_i is the on-site renewable power cost in data center i expressed in dollars per kilowatt-hour (\$/kWh), ζ_i is the electricity price in data center i expressed in dollars per kilowatt-hour (\$/kWh), C_i is the carbon footprint per unit of power used from the Grid in data center i expressed in tons of carbon per kWh (t/kWh), and α_i is the cost per unit of carbon (\$/t). Note that η_i includes the upfront investment, maintenance, and operational costs.

Finally, the total embedding cost of request j in all available data centers can be written as follows:

$$\mathcal{D}^j = \sum_{i \in V} \mathcal{D}_i^j. \tag{12}$$

The cost of embedding in the backbone network. Virtual links between the VMs that have been assigned to different data centers should be embedded in the backbone network. Let $\mathcal{P}^j$ denote the cost incurred by the InP to accommodate those virtual links. We assume that it is proportional to their bandwidth requirements and the length of physical paths to which they are mapped. It is given by

$$\mathcal{P}^j = \sum_{e' \in E^j} \sum_{e \in E} f_{e,e'} \times bw(e') \times c_p, \tag{13}$$

where c_p is the cost incurred by the InP per unit of bandwidth allocated in the backbone network.

The above embedding problem can be seen as a combination of the bin-packing problem and the multi-commodity flow problem, which are both known to be $\mathcal{NP}$-hard. In addition, to use an ILP solver, one should know the embedding costs of all possible partitions of the VDC graph in all data centers. This means that each local controller has to provide the central management framework with the embedding cost of every possible partition. This may result in a large computational overhead not only at local controllers, but also at the central controller since the number of possible partitions can be significant, especially for large-scale VDC requests. Therefore, a solution that is both efficient and scalable is required.

In the next section, we present our solution that, first, divides the VDC request into partitions such that the inter-partition bandwidth is minimized. Note that minimizing the interpartition bandwidth aims at reducing the bandwidth usage within the backbone network. Once, the partitioning is completed, we, then, use a greedy algorithm that places the obtained partitions in data centers based on location constraints and embedding costs that consider energy consumption, carbon footprint, electricity prices and PUEs of the different facilities. Finally, the algorithm optimally connects them through virtual links across the backbone network.

5 VDC Partitioning and Embedding

As mentioned earlier, our solution consists of two stages: 1) VDC partitioning, and 2) partition embedding. In the following, we present these two stages.

5.1 VDC Partitioning

Before starting the embedding process, the VDC partitioning module splits the VDC request into partitions such that the interpartition bandwidth is minimized. This allows to minimize the bandwidth usage inside the backbone network.

Our motivation stems from three main observations: 1) the cost of the inter-data center network accounts for 15 percent of the total cost, which is much higher than the cost of the intra-data center network [7], 2) wide-area transit bandwidth is more expensive than building and maintaining the internal network of a data center [26], and 3) the inter-data center network might become a bottleneck, which will eventually reduce the acceptance ratio of VDC requests. Hence, to reduce the operational costs and avoid inter-data center eventual bottleneck, it is highly recommended to reduce the inter-data center traffic [8].

The VDC partitioning problem reduces to the weighted graph partitioning problem, which is known to be $\mathcal{NP}$-Hard [27]. Hence, we propose to use the Louvain algorithm [28]. We chose the Louvain algorithm because it is a heuristic algorithm that determines automatically the number of partitions and has low time complexity of *O(n log n)*. Furthermore, it is shown to provide good results [28].

The objective of the Louvain algorithm is to maximize the modularity, which is defined as an index between -1 and 1 that measures the intrapartition density (i.e., the sum of the links' weights inside partitions) compared to the interpartition density (sum of the weights of links between partitions). In fact, graphs with high modularity have dense connections (i.e., high sum of weights) between the nodes within partitions, but sparse connections across partitions.

In a nutshell, the original Louvain algorithm proceeds as follows. Initially, every node is considered as a partition. The algorithm then considers each node and tries to move it into the same partition as one of its neighbors. The neighboring node is chosen such that the gain in modularity is maximal. Then a new graph is built by considering the partitions found during the first phase as nodes and by collapsing interpartitions links into one link (the weight of the new link is equal to the sum of the original links' weights). The same process is applied recursively to the new graph until no improvement in the modularity is possible. For more details on the original version of the Louvain algorithm, refer to [28].

However, one should note that this algorithm is not directly applicable to the VDC partitioning problem since it does not take into account location constraints.

Indeed, two VMs with two different location constraints should not be assigned to the same data center, and hence they have to belong to different partitions. However, the Louvain algorithm may not separate them, which results in nonfeasible solutions. To address this limitation, we modified the Louvain algorithm to take into account location constraints in the partitioning process. The resulting heuristic algorithm, called Location-Aware Louvain Algorithm (LALA) is described in Algorithm 1. Basically, LALA prevents moving a node from one partition to another whenever the location constraint could be violated.

Algorithm 1. Location-Aware Louvain Algorithm (LALA).

1: **IN:** $G^j(V^j, E^j)$: The VDC request to partition
2: **repeat**
3: Put every edge of G in a single partition
4: Save the initial modularity
5: **while** Nodes moved between partitions **do**
6: **for all** $v \in G^j$ **do**
7: Find the partition P such as if we move v from its partition to P:
8: -Get a maximum modularity increase
9: -There will not be two nodes with different location constraints in P
10: **if** such a partition P exists **then**
11: Move v to the partition P
12: **end if**
13: **end for**
14: **end while**
15: **if** current modularity > initial modularity **then**
16: $End \leftarrow false$
17: Change G^j to be the graph of partitions
18: **else**
19: $End \leftarrow true$
20: **end if**
21: **until** *End*

Note that, unlike previous approaches in the literature, where the number of partitions is known [10] or based on star-shaped structures detection [29], LALA determines the number of partitions as well as the shape and size of the partitions based on the modularity.

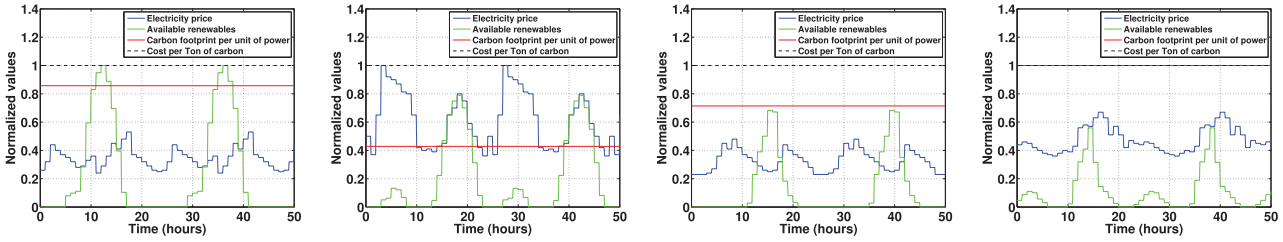
Once the VDC partitioning is completed, the second step is to assign the partitions to the data centers in such a way to minimize the operational costs as well as the carbon footprint, and provision virtual links across the backbone network to connect them. In what follows, we describe the partition placement algorithm.

5.2 Partition Embedding Problem

Once a request $G^j(V^j, E^j)$ is partitioned, the resulting partitions that are connected through virtual links can be seen as a multigraph $G^j_M(V^j_M, E^j_M)$, where V^j_M is the set of nodes (partitions) and E^j_M is the set of virtual links connecting them. The next step is to embed this multigraph in the infrastructure.

Note that at this stage, we can use the ILP formulation introduced in Section 4 by replacing the VDC request G^j by its graph of partitions G^j_M. However, even if the VDC partitioning process significantly reduces the number of components (partitions rather than VMs) to be embedded, the above formulated ILP is still NP-hard. Therefore, we propose a simple yet efficient heuristic algorithm to solve the ILP problem.

Algorithm 2 describes the proposed partition embedding algorithm. For each partition $v \in V^j_M$, we build the list of data centers able to host it based on the location constraints (lines 6-8). The idea is to start by assigning the location-constrained partitions first then select the most cost effective data centers that satisfy these constraints. For each partition $v \in V^j_M$ to embed, the central management entity queries the local controller of each data center s that satisfies the



(a) New York (b) California (c) Illinois (d) Texas

Fig. 3. Available renewables, electricity price, carbon footprint per unit of power, and cost per unit of carbon in the data centers.

location constraints to get the embedding cost of v. The cost is returned by the remote call $getCost(s, v)$, which includes both power and carbon footprint costs as described in (11). The next step is to select the data center that will host the partition v (lines 10-14). The selected data center is the one that incurs the lowest embedding cost (provided by the procedure $getCost(s, v)$), and where it is possible to embed virtual links between v and all previously embedded partitions (denoted by $N(v)$). Hence, the requirements of all virtual links in terms of bandwidth and delay are satisfied (achieved when $LinksEmbedPossible(s, v) = true$). Furthermore, links between the partition v and other partitions assigned to different data centers are embedded in the backbone network using the shortest path algorithm (lines 19-21).

Algorithm 2. Greedy VDC embedding across data centers.

1: **IN:** $G(V \cup W, E)$, $G_M^j(V_M^j, E_M^j)$
2: **OUT:** Assign each partition in V_M^j to a data center, embed the links between the partitions assigned to different data centers in the backbone network
3: **for all** $i \in V$ **do**
4: $\quad ToDC[i] \leftarrow \{\}$
5: **end for**
6: **for all** $v \in V_M^j$ **do**
7: $\quad S_v \leftarrow \{i \in V / i \text{ satisfies the location constraint}\}$
8: **end for**
9: **for all** $v \in V_M^j$ **do**
10: $\quad i \leftarrow s \in S_v$ with the smallest cost $getCost(s, v)$, and $LinksEmbedPossible(s, v) = true$
11: $\quad$ **if** no data center is found **then**
12: $\quad\quad$ **return** $FAIL$
13: $\quad$ **end if**
14: $\quad ToDC[i] \leftarrow ToDC[i] \cup \{v\}$
15: $\quad$ **for all** $k \in N(v)$ **do**
16: $\quad\quad$ **if** $k \in ToDC[i]$ **then**
17: $\quad\quad\quad ToDC[i] \leftarrow ToDC[i] \cup \{e_{vk}\}$
18: $\quad\quad$ **else**
19: $\quad\quad\quad$ **if** $\exists l \neq i \in V \ / \ k \in ToDC[l]$ **then**
20: $\quad\quad\quad\quad$ Embed e_{vk} in G using the shortest path
21: $\quad\quad\quad$ **end if**
22: $\quad\quad$ **end if**
23: $\quad$ **end for**
24: **end for**
25: **return** $ToDC$

If the whole multigraph is successfully embedded, Algorithm 2 provides the mapping of all the partitions to the data centers as well as the mapping of the virtual links that connect them in the backbone network. The complexity of this algorithm is $O(|V_M^j| \times |V|)$, where $|V_M^j|$ is the number of partitions and $|V|$ is the number of data centers.

6 Performance Evaluation

To evaluate the performance of Greenhead, we run extensive simulations using realistic topology and parameters. In the following, we present the setting of the conducted simulations, the performance metrics that we evaluated as well as the obtained results.

6.1 Simulation Settings

6.1.1 Physical Infrastructure

We consider a physical infrastructure of four data centers situated in four different states: New York, Illinois, California, and Texas. The data centers are connected through the NSFNet topology as a backbone network [30]. NSFNet includes 14 nodes located at different cities in the United States. Each data center is connected to the backbone network through the closest node to its location. We assume all NSFNet links have the same capacity of 10 Gbps [8], [31]. As illustrated in Fig. 3, the electricity price, the available renewable energy, and the carbon footprint per unit of power drawn from the Grid not only depends on the location but are also subject to change over time.

In our experiments, we simulate two working days (i.e., 48 hours). We use electricity prices reported by the US Energy Information Administration (EIA) in different locations [32]. The amount of power generated during two days are extracted from [33]. To evaluate the carbon footprint generated at each data center, we use the values of carbon footprint per unit of power provided in [34]. We also use real solar and wind renewable energy traces collected from different US states [33], and considered the on-site renewable power cost to be $\eta_i = 0.01/kWh, \forall i$ [35], [36]. To evaluate PUEs of the different data centers, we adopted the technique described in [37].

6.1.2 VDC Requests

In our simulations, similarly to previous works [4], [11], [12], [13], [16], VDCs are generated randomly according to a Poisson process with arrival rate λ. Their lifetime follows an exponential distribution with mean $1/\mu$. This mimics a real cloud environment, where VDCs could be allocated for a particular lapse of time depending on the SP requirements. This is the case for Amazon EC2, for example, where a SP can dynamically create VMs and use them only for a

specific duration. The number of VMs per VDC is uniformly distributed between 5 and 10 for small-sized VDCs and 20 and 100 for large-sized VDCs. Two VMs belonging to the same VDC are directly connected with a probability 0.5 with a bandwidth demand uniformly distributed between 10 and 50 Mbps and a delay uniformly distributed between 10 and 100 milliseconds. In addition, in each VDC, a fraction of VMs, denoted by $P_{loc} \in [0, 1]$, is assumed to have location constraints.

6.1.3 The Baseline Approach

Since, previous proposals on virtual network embedding and VDC embedding are not directly applicable to the studied scenario (see Section 2), we developed a baseline embedding algorithm that does not consider VDC partitioning. The baseline algorithm maps a VDC to the physical infrastructure by embedding its VMs and links one by one. In other words, it applies the Greenhead embedding algorithm, while considering each single VM as a partition.

6.1.4 The Simulator

We developed a C++ discrete event simulator for the central and local controllers, consisting of about 3,000 lines of code. The exchange between the central controller and the local controllers is implemented using remote procedure calls. The results are obtained over many simulation instances for each scenario, with a margin of error less than 5 percent, then we calculate the average value of performance metrics. We do not plot confidence intervals for the sake of presentation.

6.1.5 Performance Metrics

To compare our approach to the baseline, we evaluate several performance metrics including the acceptance ratio, the revenue, energy costs, the carbon footprint, and the backbone network utilization. In particular, the acceptance ratio is defined as the ratio of the number of embedded VDCs to the total number of received VDCs (i.e., including embedded and rejected VDCs). It is given by

$$\mathcal{A}_t = \frac{U_t}{N_t}, \tag{14}$$

where U_t and N_t are the number of VDC requests that have been embedded and the total number of received VDCs up to time t, respectively. The instantaneous revenue at a particular time t is given by

$$\mathcal{R}(t) = \sum_{j \in Q(t)} \mathcal{R}^j, \tag{15}$$

where $Q(t)$ is the set of VDC requests embedded in the infrastructure at time t and $\mathcal{R}^j$ as defined in (1). The cumulative revenue up to time t, denoted by $\mathcal{CR}(t)$, can then be written as

$$\mathcal{CR}(t) = \int_0^t \mathcal{R}(x)\, \mathrm{d}x. \tag{16}$$

The instantaneous power, carbon footprint and backbone network cost is given by

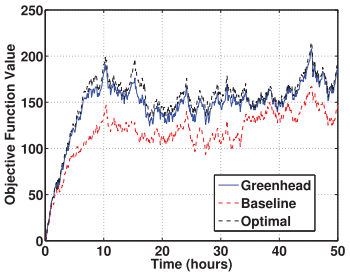


Fig. 4. Cumulative objective function obtained with Greenhead, the baseline, and the ILP solver.

$$\mathcal{C}(t) = \sum_{j \in Q(t)} \mathcal{D}_t^j + \mathcal{P}^j, \tag{17}$$

where $\mathcal{D}_t^j$ is defined in (12). Note that we add the time slot in the subscript to the definition of the $\mathcal{D}_t^j$ since we are considering the variations between different time slots. The cumulative cost up to time t can be written as

$$\mathcal{CC}(t) = \int_0^t \mathcal{C}(x)\, \mathrm{d}x. \tag{18}$$

Naturally, the instantaneous and cumulative profits are given by the difference between the instantaneous revenue and cost, and the cumulative revenue and cost, respectively.

Finally, to compare Greenhead resource allocation scheme to other schemes, we define the cumulative objective function at time t as the sum of objective function values associated to the VDCs embedded at that time. It can be written as

$$\mathcal{B}(t) = \sum_{j \in Q(t)} (\mathcal{R}^j - (\mathcal{D}^j + \mathcal{P}^j)), \tag{19}$$

where $\mathcal{R}^j - (\mathcal{D}^j + \mathcal{P}^j)$ is the objective function score of embedding VDC j as defined in (2).

6.2 Simulation Results

Through extensive experiments, we first show the effectiveness of our framework in terms of time complexity, acceptance ratio, revenue, and backbone network utilization. Then, we study the utilization of available renewable energy in the different data centers. Finally, we investigate the carbon footprint and we discuss how to spur development of green infrastructure.

6.2.1 Greenhead Provides Near-Optimal Solution within a Reasonable Time Frame

First, we compare Greenhead to an optimal solution provided by an ILP solver, as well as to the baseline in terms of computational time and solution quality (i.e., cumulative objective function). In our first set of simulations, we fixed the arrival rate λ to eight requests per hour, the average lifetime $1/\mu$ to 6 hours and the fraction of location-constrained VMs P_{loc} to 0.15. The experiments were conducted on a machine with a 3.4-GHz dual core processor and 4-GB RAM running Linux Ubuntu. To compute the optimal solution, we developed a C++ implementation of the branch-and-bound algorithm.

Fig. 4 compares the cumulative objective function (19) of the aforementioned algorithms for small-sized VDC

TABLE 2
Computation Time for Greenhead, the Baseline, and the ILP Solver (in Milliseconds)

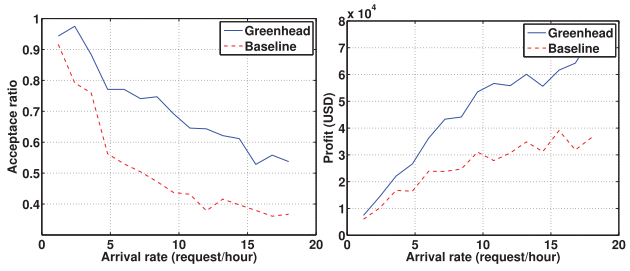
VDC size	Greenhead			Baseline	ILP Solver
	Partitioning	Embedding	Total		
5-10 VMs	0.214	0.061	0.275	0.079	13540
20-100 VMs	31.41	0.28	31.69	2.2	-

requests consisting of fully connected 5-10 VMs. We can observe that the mean values obtained for Greenhead are very close or even overlap with the values obtained with the ILP solver. This means that the Greenhead approach provides a solution close to the optimal one. We can also see that Greenhead improves the cumulative objective function value by up to 25 percent compared to the baseline.

Table 2 reports the average computation time needed to partition and embed a VDC request. The results show that Greenhead takes a very short time to partition and embed a VDC request (less than one millisecond for small-sized VDCs and up to 31 millisecond for larger VDCs). On the other hand, the ILP solver takes more than 13 seconds for small-sized VDCs. The Baseline, however, needs the least computation time since no partitioning is performed. Note that, the results for the optimal solution in large-sized VDCs were not reported since the solver was not able to find the optimal solution due to memory outage.

6.2.2 Improve Backbone Network Utilization, Acceptance Ratio, and Revenue

In the second set of experiments, we compare Greenhead to the baseline approach in terms of acceptance ratio, instantaneous revenue and backbone network utilization. To do so, we, first, fixed the arrival rate λ to eight requests per hour, the average lifetime $1/\mu$ to 6 hours and the fraction of location-constrained VMs P_{loc} to 0.15, and we simulated the infrastructure for 48 hours. Results are illustrated in Fig. 5. From this figure, we can see that Greenhead achieves, on average, 40 percent higher acceptance ratio than the baseline (see Fig. 5a) and up to 100 percent more instantaneous profit (see Fig. 5b). Although both schemes lead to almost the same utilization of the backbone network on average (see Fig. 5c), they differ in the fact that Greenhead avoids embedding virtual links with high bandwidth demand in the backbone network thanks to the partitioning algorithm. Hence, it ensures that the embedded requests consume as less bandwidth as possible inside the backbone network. This is confirmed by Fig. 5d,



(a) Acceptance Ratio (b) Profit

Fig. 6. Acceptance ratio and revenue for different arrival rates ($P_{loc} = 0.10$).

which compares the average used bandwidth per request inside the backbone network for both schemes. It is clear that requests embedded by Greenhead use on average 40 percent less bandwidth in the backbone network than the baseline algorithm.

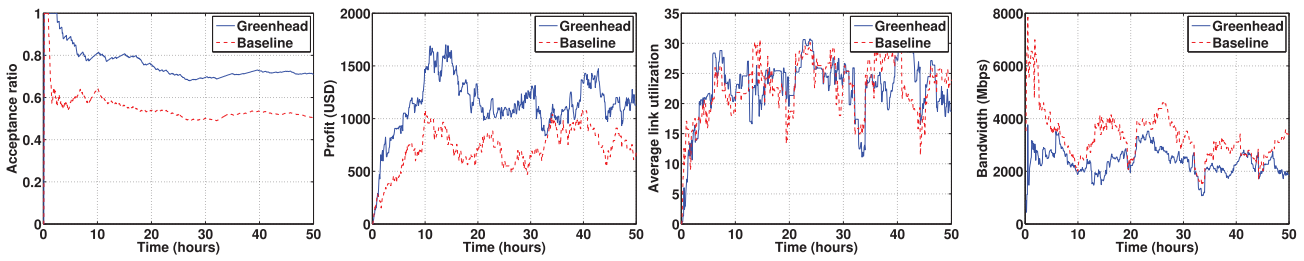
Figs. 6 and 7 show the performance results when varying the arrival rate λ and P_{loc}, respectively.

From Fig. 6, we can notice that as the arrival rate increases, more requests are embedded, which results in higher revenue. At the same time, the acceptance ratio goes down since there is no room to accept all the incoming requests. It is also clear from this figure that the acceptance ratio as well as the revenue are always higher for Greenhead compared to the baseline.

However, this benefit is reduced when $P_{loc} = 0$ as shown in Fig. 7. In fact, when there are no location constraints, the VDCs can be hosted in any data center, and hence, their placement is only driven by the availability of renewables, the electricity price, and the carbon footprint. In practice, if the data centers are not overloaded, any particular VDC is entirely hosted in the same data center. This results in low backbone network utilization as shown in Fig. 7c. On the other hand, when $P_{loc} = 1$, all the VMs have to be placed as required by the SP. As a result, the Greenhead is not able to perform any optimization. Finally, when the fraction of the constrained VMs is between 0 and 1, the Greenhead has more freedom to decide of the nonconstrained VMs placement. In this case, Greenhead is able to optimize VDCs allocation and significantly improve the acceptance ratio and revenue compared to the baseline.

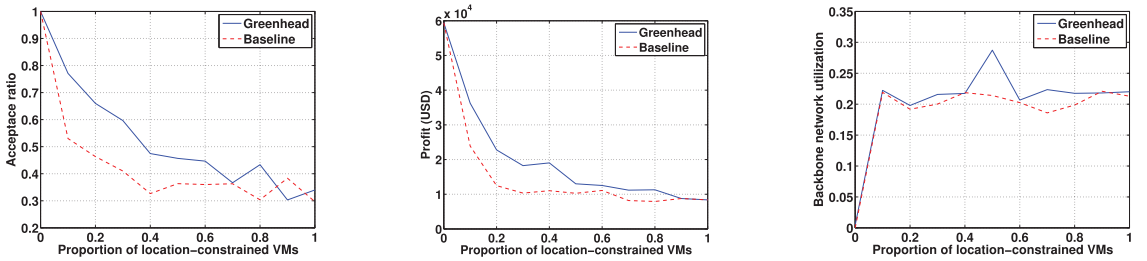
6.2.3 Maximize Renewables' Usage

To illustrate how our proposed framework exploits the renewables in the different data centers, we studied the power consumption across the infrastructure and particularly the usage of renewable energy. Fig. 8 shows the total



(a) Acceptance ratio (b) Instantaneous profit (c) Average link utilization in the backbone network (d) Used bandwidth per request in the backbone network

Fig. 5. Greenhead versus the baseline. ($\lambda = 8$ requests/hour, $1/\mu = 6$ hours, $P_{loc} = 0.15$, duration = 48 hours.)



(a) Acceptance Ratio (b) Instantaneous profit (c) Average link utilization in the backbone network

Fig. 7. Impact of the fraction of location-constrained VMs. ($\lambda = 8$ requests/hour.)

power consumption across the infrastructure for both Greenhead and the baseline approach. It is clear from this figure that Greenhead consumes much more power than the baseline since it accepts more VDC requests. We can also see that it uses up to 30 percent more renewable power than the baseline.

Fig. 9 shows the impact of the fraction of location-constrained VMs on the power consumption across the infrastructure. We can notice that, as the fraction of constrained nodes increases, Greenhead uses more power from the Grid. For instance, with $P_{loc} = 0$, Greenhead uses 100 percent of available renewables. However, when P_{loc} is getting higher, up to 15 percent of the available renewables are not used. This is due to the fact that the VMs with location constraints can only be embedded in specific data centers, which may not have available renewables. Consequently, more power is drawn from the Grid.

6.2.4 Reduce Energy Consumption and Carbon Footprint Per Request

Fig. 10 compares the obtained results for both schemes for all studied metrics. We can observe that Greenhead improves up to 40 percent the acceptance ratio which translates into 48 percent more profit. Furthermore, Greenhead uses up to 15 percent more renewables and reduces the consumed power per request by 15 percent compared to the baseline approach. In addition, we can notice that, while Greenhead boosts significantly the profit up to 48 percent, it generates the same amount of carbon footprint compared to the baseline approach.

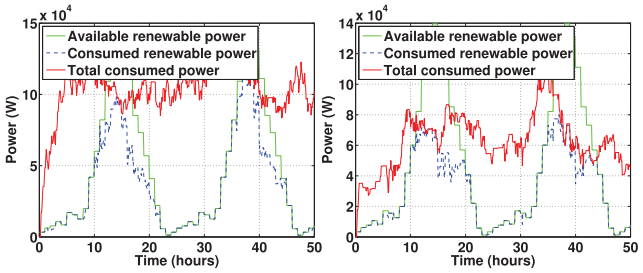
6.2.5 Green Infrastructure Is Possible Through Tuning, at the Expense of Power Cost

Finally, Fig. 11 shows the impact of varying the cost per unit of carbon ($\alpha_i = \alpha,\ \forall i \in V$) on the carbon footprint in the whole infrastructure as well as the total power cost. In this experiment, λ is set equal to 8 request/hour and P_{loc} equal to 0.1. From this figure, we can see that a tradeoff between the carbon footprint and the power cost can be achieved. In addition, we can notice that an InP can set a carbon footprint target to reach by choosing the corresponding value of α. For instance, one can reduce the carbon footprint by 12 percent while increasing the power cost by only 32 percent by setting α to $80\ \$/t$.

It is worth noting that nowadays, the carbon cost is imposed by governments as a carbon tax whose cost is between 25 and 30 $ [38], [39], [40]. According to Fig. 11, such a cost is not enough to force InPs to reduce their carbon footprint.

To explain the power cost increase when reducing the carbon footprint, let us explore Fig. 12, which presents the power consumption in different data centers. From this figure, we can notice that for small values of α (i.e., $\alpha \leq 160\ \$$), Greenhead uses more the data centers in Illinois and New York. These two data centers have low electricity prices (see Fig. 3) but high carbon footprint (0.0006 ton/Kwh and 0.0005 ton/Kwh, respectively). However, as α increases, Greenhead uses the data center in California since it has the smallest carbon footprint per unit of power (0.0003 ton/Kwh), but a higher electricity price (on average, 100 percent higher compared to New York data center).

Consequently, we can conclude that: 1) to reduce data centers' carbon footprint, governments should consider much higher carbon taxes, and 2) using Greenhead, a socially-responsible InP should consider higher carbon costs, even by artificially increasing these costs, to force Greenhead to use environment-friendly data centers to reduce the carbon footprint.

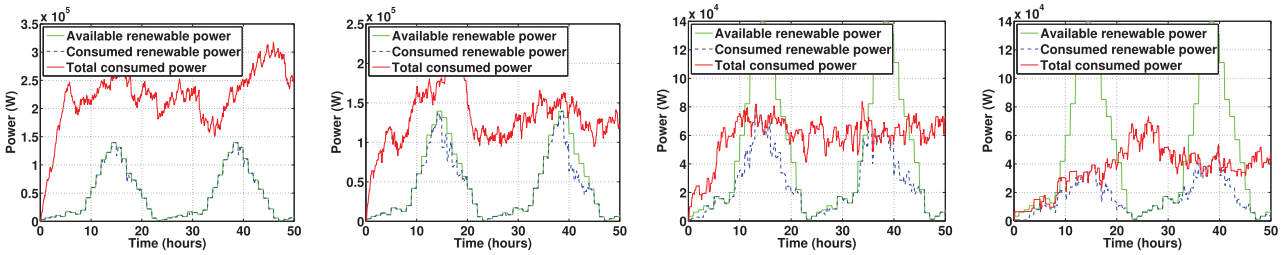


(a) Greenhead (b) The baseline

Fig. 8. Power consumption across the infrastructure ($\lambda = 8$ requests/hour, $P_{loc} = 0.20$).

7 Conclusions

The last few years witnessed a massive migration of businesses, services, and applications to the cloud. Cloud providers take advantage of the worldwide market to deploy geographically distributed infrastructures and enlarge their coverage. However, multiple data centers consume massive amounts of power. Furthermore, their carbon footprint is a rapidly growing fraction of total emissions. In this paper, we proposed Greenhead, a holistic resource management framework for embedding VDCs across a geographically distributed infrastructure. The goal of Greenhead is to find the best tradeoff between maximizing revenue, reducing energy costs, and ensuring the environmental friendliness of the infrastructure. The key idea of the proposed solution is to conquer the complexity of the problem by partitioning the VDC request



(a) $P_{loc} = 0.0$ (b) $P_{loc} = 0.1$ (c) $P_{loc} = 0.6$ (d) $P_{loc} = 1$

Fig. 9. The utilization of the renewables in all data centers for different fractions of location-contained nodes P_{loc} for Greenhead ($\lambda = 8$ requests/hour).

based on the bandwidth requirements between the VMs. The partitions and the virtual links connecting them are then dynamically assigned to the infrastructure data centers and backbone network to achieve the desired objectives.

We conducted extensive simulations for four data centers connected through the NSFNet topology. The results show that Greenhead provides near-optimal solution within a reasonable computational time frame and improves requests' acceptance ratio and InP revenue by up to 40 percent while ensuring high usage of renewable energy and minimal footprint per request.

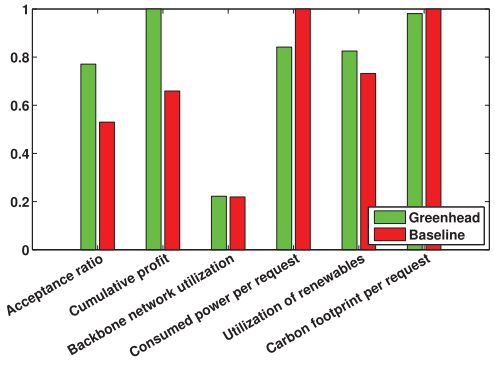


Fig. 10. Comparison of the average values of the different metrics.

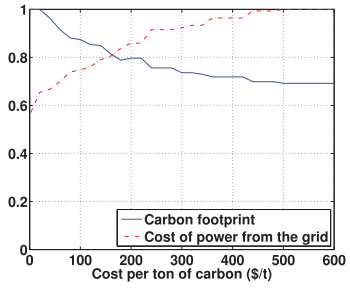


Fig. 11. The carbon footprint (normalized values) of the whole infrastructure with variable cost per ton of carbon.

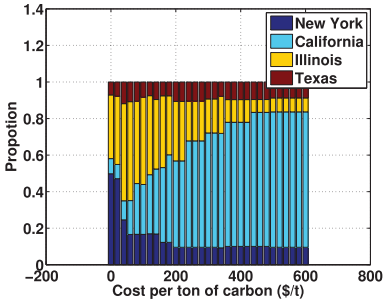


Fig. 12. The power from the Grid (normalized values) used in different data centers with variable cost per ton of carbon α.

ACKNOWLEDGMENTS

This work has been supported in part by the NSERC Discovery program and in part by the NSERC Smart Applications on Virtualized Infrastructures (SAVI) Research Network.

REFERENCES

[1] *Amazon Elastic Compute Cloud (Amazon EC2)*, http://aws.amazon.com/ec2/, 2013.
[2] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Toward Predictable Datacenter Networks," *Proc. ACM SIGCOMM*, pp. 242-253, 2011.
[3] M.F. Bari, R. Boutaba, R. Esteves, Z.G. Lisandro, M. Podlesny, G. Rabbani, Q. Zhang, and M.F. Zhani, "Data Center Network Virtualization: A Survey," 2012.
[4] M.F. Zhani, Q. Zhang, G. Simon, and R. Boutaba, "VDC Planner: Dynamic Migration-Aware Virtual Data Center Embedding for Clouds," *Proc. IFIP/IEEE Integrated Network Management Symp. (IM 2013)*, May 2013.
[5] C. Guo, G. Lu, H.J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "SecondNet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees," *Proc. Sixth Int'l Conf. (Co-NEXT)*, pp. 1-12, 2010.
[6] A. Vahdat, "SDN Stack for Service Provider Networks," *Proc. Open Networking Summit*, 2012.
[7] A. Greenberg, J. Hamilton, D.A. Maltz, and P. Patel, "The Cost of a Cloud: Research Problems in Data Center networks," *ACM SIGCOMM Computer Comm. Rev.*, vol. 39, no. 1, pp. 68-73, Dec. 2008.
[8] Forrester Research, "The Future of Data Center Wide-Area Networking," 2010.
[9] ITU, "Toolkit on Environmental Sustainability for the ICT Sector (ESS)," http://www.itu.int/ITU-T/climatechange/ess/index.html, 2012.
[10] Y. Xin, I. Baldine, A. Mandal, C. Heermann, J. Chase, and A. Yumerefendi, "Embedding Virtual Topologies in Networked Clouds," *Proc. Sixth Int'l Conf. Future Internet Technologies (CFI '11)*, pp. 26-29, 2011.
[11] M. Chowdhury, M. Rahman, and R. Boutaba, "Vineyard: Virtual Network Embedding Algorithms with Coordinated Node and Link Mapping," *IEEE/ACM Trans. Networking*, vol. 20, no. 1, pp. 206-219, Feb. 2012.
[12] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual Network Embedding through Topology-Aware Node Ranking," *ACM SIGCOMM Computer Comm. Rev.*, vol. 41, no. 2, pp. 38-47, Apr. 2011.
[13] I. Fajjari, N. Aitsaadi, G. Pujolle, and H. Zimmermann, "VNE-AC: Virtual Network Embedding Algorithm Based on Ant Colony Meta-heuristic," *Proc. IEEE Int'l Conf. Comm. (ICC)*, pp. 1-6, June 2011.
[14] J.F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. de Meer, "Energy Efficient Virtual Network Embedding," *IEEE Comm. Letters*, vol. 16, no. 5, pp. 756-759, May 2012.

[15] S. Su, Z. Zhang, X. Cheng, Y. Wang, Y. Luo, and J. Wang, "Energy-Aware Virtual Network Embedding through Consolidation," *Proc. IEEE INFOCOM,* pp. 127-132, 2012.
[16] I. Houidi, W. Louati, W.B. Ameur, and D. Zeghlache, "Virtual Network Provisioning across Multiple Substrate Networks," *Computer Networks,* vol. 55, no. 4, pp. 1011-1023, Mar. 2011.
[17] M. Chowdhury, F. Samuel, and R. Boutaba, "Polyvine: Policy-Based Virtual Network Embedding across Multiple Domains," *Proc. ACM SIGCOMM,* pp. 49-56, 2010.
[18] Q. Zhang, Q. Zhu, M.F. Zhani, and R. Boutaba, "Dynamic Service Placement in Geographically Distributed Clouds," *Proc. IEEE 32nd Int'l Conf. Distributed Computing Systems (ICDCS),* pp. 526-535, June 2012.
[19] M. Adnan, R. Sugihara, and R. Gupta, "Energy Efficient Geographical Load Balancing via Dynamic Deferral of Workload," *Proc. IEEE Fifth Int'l Conf. Cloud Computing (CLOUD),* pp. 188-195, June 2012.
[20] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the Electric Bill for Internet-Scale Systems," *ACM SIGCOMM Computer Comm. Rev.,* vol. 39, no. 4, pp. 123-134, Aug. 2009.
[21] Z. Liu, M. Lin, A. Wierman, S.H. Low, and L.L. Andrew, "Geographical Load Balancing with Renewables," *SIGMETRICS Performance Evaluation Rev.,* vol. 39, no. 3, pp. 62-66, Dec. 2011.
[22] Z. Liu, M. Lin, A. Wierman, S.H. Low, and L.L. Andrew, "Greening Geographical Load Balancing," *Proc. ACM SIGMETRICS Joint Int'l Conf. Measurement and Modeling of Computer Systems (SIGMETRICS '11),* pp. 233-244, 2011.
[23] J. He, X. Deng, D. Wu, Y. Wen, and D. Wu, "Socially-Responsible Load Scheduling Algorithms for Sustainable Data Centers over Smart Grid," *Proc. IEEE Third Int'l Conf. Smart Grid Comm. (SmartGridComm),* Nov. 2012.
[24] P.X. Gao, A.R. Curtis, B. Wong, and S. Keshav, "It's Not Easy Being Green," *Proc. ACM SIGCOMM,* pp. 211-222, 2012.
[25] K. Le, J. Zhang, J. Meng, R. Bianchini, Y. Jaluria, and T. Nguyen, "Reducing Electricity Cost through Virtual Machine Placement in High Performance Computing Clouds," *Proc. Int'l Conf. High Performance Computing, Networking, Storage, and Analysis (SC),* pp. 1-12, Nov. 2011.
[26] A. Greenberg, J.R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D.A. Maltz, P. Patel, and S. Sengupta, "VL2: A Scalable and Flexible Data Center Network," *Proc. ACM SIGCOMM,* pp. 51-62, 2009.
[27] S.E. Schaeffer, "Graph Clustering," *Computer Science Rev.,* vol. 1, no. 1, pp. 27-64, 2007.
[28] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast Unfolding of Communities in Large Networks," *J. Statistical Mechanics: Theory and Experiment,* Oct. 2008.
[29] T. Ghazar and N. Samaan, "Hierarchical Approach for Efficient Virtual Network Embedding Based on Exact Subgraph Matching," *Proc. IEEE GLOBECOM,* 2011.
[30] *The Nat'l Science Foundation Network (NSFNET),* http://www.nsf.gov, 2013.
[31] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-Datacenter Bulk Transfers with Netstitcher," *Proc. ACM SIGCOMM,* pp. 74-85, 2011.
[32] *U.S. Energy Information Administration,* http://www.eia.gov, 2013.
[33] *The Renewable Resource Data Center (RReDC) Website,* http://www.nrel.gov/rredc/, 2012.
[34] *Carbon Footprint Calculator,* http://www.carbonfootprint.com, 2012.
[35] X. Deng, D. Wu, J. Shen, and J. He, "Eco-Aware Online Power Management and Load Scheduling for Green Cloud Datacenters," technical report, CS Department, Sun Yat-sen Univ., Apr. 2013.
[36] P. Heptonstall, "A Review of Electricity Unit Cost Estimates," UK Energy Research Centre Working Paper, May 2007.
[37] A. Qouneh, C. Li, and T. Li, "A Quantitative Analysis of Cooling Power in Container-Based Data Centers," *Proc. IEEE Int'l Symp. Workload Characterization,* 2011.
[38] "Carbon Taxation and Fiscal Consolidation, The Potential of Carbon Pricing to Reduce Europe's Fiscal Deficits," Report prepared for the European Climate Foundation and Green Budget, http://bit.ly/L7i3td, May 2012.
[39] Climate Action Plan Tax, http://bit.ly/XyGk32, June 2010.
[40] British Columbia Carbon Tax, http://bit.ly/JLUurv, Feb. 2008.

Ahmed Amokrane received the graduation degree with honors from the Ecole Nationale Supérieure dInformatique (ESI), Algeria, and Ecole Normale Supérieure de Cachan (ENS Cachan), France, the engineering degree in computer science from ESI in 2010, and the MSc degree in computer science from ENS Cachan, in 2011, respectively. He is currently working toward the PhD degree at the University of Pierre and Marie Curie-Paris 6, France. His research interests include energy efficient and green networking in wireless and data center networks, resource management in wireless mesh and cloud computing environments and software defined networking. He is a student member of the IEEE.

Mohamed Faten Zhani received the engineering and MS degrees from the National School of Computer Science, Tunisia in 2003 and 2005, respectively, and the PhD degree in computer science from the University of Quebec in Montreal, Canada, in 2011. Since then, he has been a postdoctoral research fellow at the University of Waterloo. His research interests include virtualization, resource management in cloud computing environments, network performance evaluation, and software-defined networking. He is a member of the IEEE.

Rami Langar received the MS degree in network and computer science from the University of Pierre and Marie Curie—Paris 6, in 2002, and the PhD degree in network and computer science from Telecom ParisTech, France, in 2006. He is currently an associate professor at the LIP6, University of Pierre and Marie Curie—Paris 6, France. In 2007 and 2008, he was with the School of Computer Science, University of Waterloo, Ontario, Canada, as a postdoctoral research fellow. His research interests include mobility and resource management in wireless mesh, vehicular ad hoc and femtocell networks, performance evaluation, and quality-of-service support. He is a member of the IEEE.

Raouf Boutaba received the MSc and PhD degrees in computer science from the University Pierre and Marie Curie, Paris, in 1990 and 1994, respectively. He is currently a professor of computer science at the University of Waterloo and a distinguished visiting professor at the division of IT convergence engineering at POSTECH. His research interests include network, resource, and service management in wired and wireless networks. He is the founding editor in chief of the *IEEE Transactions Network and Service Management* (2007-2010) and on the editorial boards of other journals. He has received several Best Paper Awards and other recognitions such as the Premiers Research Excellence Award, the IEEE Hal Sobol Award in 2007, the Fred W. Ellersick Prize in 2008, and the Joe LociCero and the Dan Stokesbury awards in 2009. He is a fellow of the IEEE.

Guy Pujolle received the PhD degree from the University of Paris IX and "Thse d'Etat" degrees in computer science from the University of Paris XI, in 1975 and 1978, respectively. He is currently a professor at Pierre et Marie Curie University—Paris 6, a distinguished invited professor at POSTECH, Korea, a member of the Institut Universitaire de France, and a member of The Royal Physiographical Academy of Lund, Sweden. From 1994 to 2000, he was a professor and head of the Computer Science Department of Versailles University. He was also a professor and head of the MASI Laboratory at Pierre et Marie Curie University (1981-1993), Professor at ENST (1979-1981), and a member of the scientific staff of INRIA (1974-1979). He is the French representative at the Technical Committee on Networking at IFIP. He is an editor for*ACM International Journal of Network Management, Telecommunication Systems*, and editor-in-chief of *Annals of Telecommunications*. He is a pioneer in high-speed networking having led the development of the first Gbit/s network to be tested in 1980. He was participating in several important patents like DPI or virtual networks. He is the cofounder of QoSMOS (www.qosmos.fr), Ucopia Communications (www.ucopia.com), EtherTrust (www.ethertrust.com), Virtuor (www.VirtuOR.fr), and Green Communications (www.green-communications.fr). He is a member of the IEEE.