An Energy-Aware Embedding Algorithm for Virtual Data Centers

Tran Manh Nam, Nguyen Van Huynh, Le Quang Dai, Nguyen Huu Thanh
School of Electronics and Telecommunications
Hanoi University of Science and Technology
Hanoi, Vietnam

***Abstract*—Cloud computing has emerged in the recent years as a promising paradigm that facilitates such new service models as Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). As the number of cloud service provider increases, there exists a demand to dynamically provision virtual data centers (VDC) on top of the infrastructure provider's physical data centers. This research addresses problems related to embedding virtual data centers inside physical data centers. VDC embedding is challenging as it is an *NP-hard* problem that should meet multiple objectives. In the paper, we propose a new heuristic VDC embedding algorithm that takes into account energy consumption as well as physical resources of data centers. We also realize this virtualization paradigm based on a Software-Defined Network architecture.**

I. Introduction

Cloud computing is becoming increasingly important nowadays as it supports new business models such as *Infrastructure-as-a-Service* (IaaS), *Platform-as-a-Service* (PaaS) and *Software-as-a-Service* (SaaS). One important component of cloud computing are data centers, which are used by cloud service providers to house cloud-based resources and services. In cloud computing paradigms, cloud service providers can typically build their own data centers to offer cloud services or alternatively make use of data centers provided by third-party *Infrastructure Providers* (InP). In either former and later case, *data center virtualization* comes into play, which is a concept of *network virtualization* (NV) that allows creating multiple, separated virtual data centers (VDC) on top of physical data centers. Creating VDCs on top of the physical cloud substrates has the following advantages:

- *Cost saving*: data center virtualization allows reducing capital expenditure (CAPEX) and operational costs (OPEX) as several cloud service providers can share the same physical data center of a third-party infrastructure provider.
- *Energy saving*: Recent surveys have shown that the energy consumption in a data center considerably contributes to its operation costs. A remarkable part of the large energy volume consumed in data centers today is due to the over-provisioning of such network resources as switches, links, and servers to meet the stringent requirements on reliability. By dynamically scale up and down the VDC instead of maintaining a fixed number of physical servers, the under-utilization of servers and network can be avoided that leads to more energy-efficient usage of the data center.
- *Flexibility*: the VDC can be dynamically provisioned on demand based on service requirements, scalability, time duration and required resources.

A major challenge of network virtualization in data centers is the *virtual data center embedding* (VDCE) problem that consists of two sub-problems: (1) embedding virtual machines (VM) on physical servers in the physical data center based on the VDC embedding request (see III-A); and (2) creating a virtual network consisting of virtual switches and links interconnecting these virtual machines.

Solving the VDC embedding problem is *NP-hard*. For that reason current research mostly follows heuristic and meta-heuristic approaches. In this research, we focus on energy-efficient virtual data center embedding approaches with the following contributions:

- A novel *VDC embedding algorithm* with the following objectives: (1) *Resource efficiency* that deals with efficient mapping of virtual resources on substrate resources in terms of CPU, memory and network bandwidth; (2) *Energy efficiency* that deals with minimizing energy consumption of the virtual data center while meeting mapping demands; and (3) *Flexibility* that deals with how the VDC can be dynamically provisioned, changed and removed according to actual needs. Evaluation results show that our algorithm performs better than some existing ones in terms of acceptance ratio and energy consumption.
- A *data center virtualization architecture* based on Software-Defined Networking technology that has full control of network and server virtualization in the physical data center and allows performing VDC mapping dynamically and flexibly.

The rest of the paper is organized as follows. Section II discuss some related work on virtualization in data centers. Section III formulates the VDC embedding problems and power profiling and modeling. In Section IV, we propose a new energy-aware VDC embedding algorithm with the above objectives. Section V shows some evaluation results. The last section concludes the work.

978-0-9883045-1-2/16 $31.00 © 2016 ITC

II. Related Work

A. Virtual data center embedding

As already addressed above, one of the key challenges in building virtual data centers is the VDC embedding problem, which requires to map VDC components such as virtual machines (VM) and network devices (switches and links) onto physical servers, nodes and links. There is only few research work that has addressed the VDC embedding problem. For instance, VDC Planner [6] and Venice [9] were proposed as VDC embedding methods based on migration-aware model to maximize the revenue of InPs. Gue et al. [11] proposed a data center network architecture called SecondNet that incorporate a greedy algorithm for resource allocation to VDC. SecondNet focuses on providing bandwidth guarantees among multiple VMs in a multi-tenant virtualized data center. Amokrane et al. [8] introduced GreenHead - a holistic resource management framework for embedding VDCs across geographically distributed data centers connected through a backbone network. For energy efficiency, Han et al. [10] proposed SAVE - an SDN Assisted VDC Embedding system. However, those methods did not consider the life time of VDCs as well as arriving and leaving time of their requests. Besides, recently there is some work focusing on energy efficiency of servers by using servers consolidation and placement algorithms for VMs that can be mapped onto physical servers [1], [2], [3]. Nevertheless, these approaches only focus on a single group of VMs requests and not on the embedding of virtual data centers that include multiple groups of VM requests at a time. To the best of our knowledge, energy-efficient VDC approaches that take into account the dynamic embedding of dynamic VDC requests and address energy consumption of both physical servers and network devices are still lacking.

B. DC network – Fat-tree

DC network topology: In this work, we choose the *Fat-tree* topology [16] as the network topology of data center. One of the most advantage of this topology is the reduction of the oversubscription ratio that allows removing bottleneck point of the hierarchical architecture. A k Fat-tree is a network architecture of DC that uses the same k*-port* switches with three layer: *edge, aggregation* and *core*. Like [18] we divide the traffic pattern into three scenarios: *near, middle* and *far*. In the near traffic scenario, the source and destination of the flow are connected to the same edge switch, so that the exchanged traffic traverses over only one switch. On the other hand, a flow in the middle traffic scenario has the source/destination pair residing in the same POD but does not connect to the same edge switch, so that the traffic traverses over three switches (two edge and one aggregation switch). Finally, in far traffic scenario, the source/destination pair of flow stay on different PODs (Performance Optimized Data Centers), so that core, aggregation and edge switches are involved in the communication. Based on these scenarios, in this work, we propose three groups of servers for the embedding algorithm (discuss later in section IV).

C. Related technologies for virtualization

Despite the advances in virtualizing computing and storage elements, the network is still mostly statically configured in a box-by-box manner. One might think that long standing virtualization primitives such as VLANs (virtualized L2 domain), NAT (virtualized IP address space), and MPLS (virtualized path) are enough to provide full and automated network virtualization. However, there is no single unifying abstraction that can be leveraged to configure (or reconfigure) the network in a global manner. As a consequence, current network provisioning can take long, while computing provisioning takes only minutes [20]. Optimizing and realizing Network Virtualization has attracted much attention of research communities and important role in data center virtualization at this time.

There are many approaches of network virtualization that have already been under research and used for the future of the Internet testbeds [4], [5]. Network virtualization provides an abstraction of coexistence of multiple virtual networks on the same physical substrate network. In data center contexts, NV should cooperate with virtual machine consolidation. The introduction of Software-Defined Networking, an emerging networking paradigm that gives hope to change the limitations of current network infrastructures, has made network virtualization more easily. The network virtualization concept in the context of Software-Defined Networking was first mentioned by Sherwood et al. [21]. The authors propose a framework called FlowVisor which allows the same hardware forwarding plane to be shared among multiple logical networks, each with distinct forwarding logic. In [22] we extend FlowVisor to multilayer NV with resource reservation. Software-Defined Networking [20] is an emerging architecture of the future Internet that is programmable, flexible and centralized manageable, making it ideal for the high-bandwidth, dynamic nature of today's Internet services. In this paper, we propose a data center virtualization architecture based on Software-Defined Networking technology that allows performing VDC mapping flexibly.

III. Problem Formulation

A. Data center modeling

In this section, we model the virtual data center embedding as an optimization problem focusing on minimizing total power consumption of both servers and network devices spent on embedding VDC requests.

Physical DC infrastructure: We model a physical DC infrastructure as a weighted graph $G^p = (S^p, N^p, L^p)$ where S^p denotes a set of physical servers, N^p denotes a set of network devices (switches) and L^p denotes a set of physical links. For physical servers, the attributes generally include *memory* and *CPU capacity*. $M_{cap}(S^p)$ and $C_{cap}(S^p)$ denote available (or leftover) memory and CPU of S^p, respectively. For physical links, the attribute is *bandwidth* so that $B_{cap}(L_i^p)$ denotes available bandwidth of a link L_i^p. Note that the modeling, analysis and algorithm in this paper can be easily extend to incorporate other attributes.

TABLE I
TERMINOLOGY USED THROUGHOUT THIS WORK

Terms	Description
$G^p(S^p, N^p, L^p)$	Physical DC infrastructure
S^p, N^p, L^p	Set of physical machines, switches and links, respectively
$B_{cap}(L_i^p)$	Available bandwidth of physical link L_i^p
$C_{cap}(S^p)$	Available CPU of physical server S^p
$M_{cap}(S^p)$	Available memory of physical server S^p
$R_i^v(VM_i, L^{v_i}, t_i, d_i)$	VDC request i^{th} with set of virtual machines, matrix of links demand, arrival time and duration
$l_{s,d}^{v_i}$	Requested bandwidth from source vm_s to destination vm_d
M_d, C_d, B_d	Memory, CPU demand of server and bandwidth of link, respectively
$cap : S^p \cup L^p \rightarrow G^p$	Function assigns a available capacity to an element of G^p (either servers or links)
$dem_i : VM_i \cup L^{v_i} \rightarrow G^p$	Function assigns a demand to an element of VDC request R_i^v (either VM or bandwidth)
$f_i : VM_i \rightarrow S^p$	Function that maps virtual machine to physical machine (*VmM)*
$k_i : L^{v_i} \rightarrow (N^p, L^p)$	Function that maps matrix of BW demands onto a part of physical DC (*VLiM)*
P_S, P_N	Power consumption of servers and network devices

VDC request: A sequence of virtual data center requests joins and leaves over time. Similarly to G^p, we model i^{th} VDC as a weighted graph $R_i^v = (VM_i, L^{v_i}, t_i, d_i)$, in which t_i and d_i denote the arrival time and duration of VDC, respectively. VM_i denotes the a set of virtual machines with the corresponding computing resource $C_d(VM_i)$ and memory $M_d(VM_i)$ demands. L^{v_i} denotes the matrix of bandwidth demand including $l_{s,d}^{v_i} \in L^{v_i}$ that is the bandwidth demand from the source vm_s to the destination vm_d.

VDC embedding: The main challenge in creating virtual data centers is the VDC embedding problem, which maps VDCs onto the physical data center, which includes physical servers and links. Given a VDC request R_i^v and a physical data center G^p, embedding R_i^v onto G^p means to find a subset of S^p, N^p, L^p at time t_i that satisfies the requirement of VM_i and L^{v_i}. Solving this embedding problem as Integer Linear Programming is *NP-hard* [13]. In this work we divide it into two subproblems: (1) virtual VM mapping (*VmM*) that maps the VMs of VDC request onto the physical servers; and (2) virtual link mapping (*VLiM*) that maps matrix of link demands onto the substrate links. Let $cap : S^p \cup L^p \rightarrow G^p$ be a function that returns an available capacity of physical DC, either servers or network devices. Besides, for each VDC request i^{th}, let $dem_i : VM_i \cup L^{v_i} \rightarrow G^p$ be a function that assigns demand to an element of this VDC. Then, a VDC embedding consists of two functions *VmM* (Eq.1) and *VLiM* (Eq.2).

$$f_i : VM_i \rightarrow S^p \tag{1}$$

$$k_i : L^{v_i} \rightarrow (N^p, L^p) \tag{2}$$

Such that these two mapping functions form an embedding for VDC_i. Computational resources (CPU and memory) required by a vm_j must be lower than those of physical server hosting it and the required bandwidth of a virtual link must be lower than the available bandwidth of all physical links on the path of the DC that the virtual link $l_{s,d}^{v_i}$ is mapped.

$$\forall vm_i \in VM_i : dem_i(vm_i) \leq cap(f_i(vm_i)) \tag{3}$$

$$\forall l_{s,d}^{v_i} \in L^{v_i} : \forall L_i^P \in k_i(l_{s,d}^{v_i}) : dem_i(l_{s,d}^{v_i}) \leq cap(L_i^p) \tag{4}$$

Let $x_{i^k}^i$ be a binary function indicating whether the VM i^k is allocated in server i. Let $state_t : S^p \cup N^p \cup L^p \rightarrow G^p$ denote the function returning a state at time t of an element of the DC by binary values, which return 1 when turning on (ON_State) and 0 otherwise (OFF_State). Thus,

- $state(s_i, t)$ - The working state of the physical server $s_i \in S^p$ at time t.
- $state(n, t)$ - The working state of the physical network device $n \in N^p$at time t.
- $state(L_i^p, t)$ - The working state of the physical links $L_i^p \in k_i(l_{s,d}^{v_i})$ at time t

Then we have the constraints of the functions f_i and $k_i$as below.

One virtual machine is mapped on only one physical server (Eq.5) if successful or none if unsuccessful (Eq.6)

$$\sum_{i \in S^p} x_{i^k}^i = 1, \forall i^k \in VM \tag{5}$$

$$x_{i^k}^i = 0 \tag{6}$$

All physical elements that a VDC_i is allocated on must be turned on (Eq.7 and Eq.8).

$$\forall S_i^p \in f_i(VM_i) : state(S_i^p, t) = 1 \tag{7}$$

$$\forall l_{s,d}^{v_i} \in L^{v_i} : \forall L_i^p \in k_i(l_{s,d}^{v_i}) : state(L_i^p, t) = 1 \tag{8}$$

B. Energy modeling

The working state of physical machines, switches and links are formulate as follows:

1) Physical machines: Energy consumption model of physical machines is defined as Eq.(9) where st_{s_i} is binary indicator whether i^{th} server turned on (1) or turned off (0) and p_{s_i} is energy consumption.

$$P_S(t) = \sum_{\forall s_i \in S^p} state(s_i, t).p_{s_i} \tag{9}$$

2) Network devices: The working state of switches i at time t is defined by binary indicator $st^n(t)$ = 0 (switch is turned off) or 1 (otherwise). The energy consumption of the network part of DC (switches and links), $P_N(t)$, at time t is denoted as summing of all switches with static power (baseline power), P_{static}, and power consumption of interfaces under their operating speeds (see Eq.(10)).

$$P_N(t) = \sum_{\forall n \in N^p} state(n,t).[P_{static} + \sum_{j=1}^{k}(m_j.P_j)] \quad (10)$$

In this work, for power measurement of network devices, we use power profile of an energy-aware commercial 24-port switches HP [14]. The switch is able to change the clock frequencies of its network interfaces for different power states. Thus P_{static} and power consumption of ports under separate operating speeds $10Mbps, 100Mbps, 1000Mbps$ (P_j) is summarized in Table II. Besides, the average typical power consumption for a server IBM 2U rackmount x86 is 400W as described in [15].

TABLE II
POWER SUMMARY FOR A HP ENTERPRISE SWITCH

Operating speed	Power (mW)
P_{static}	39.000
P_{10} - 100Mbps per port	420
P_{100} - 1000Mbps per port	480
P_{1000} - 1Gbps per port	900

C. Energy-saving problem formulation

Basically, the main objective of this work is to reduce the total energy consumption of the physical DC. Let P_{DC} represent be the total energy consumption of all servers and network devices of the DC. The energy-aware VDC embedding algorithm focuses on decreasing the number of ON_state servers and network devices. This objective is defined as Eq.(11):

$$\min\Big(\sum_{\forall s_i \in S^p} state(s_i,t).p_{s_i} + \sum_{\forall n \in N^p} state(n,t).[P_{static} + \sum_{j=1}^{k}(m_j.P_j)]\Big) \quad (11)$$

The constraints of this VDC embedding are from Eq. (3) – Eq. (8).

D. VDC request modeling

Collecting traffic load measurements in real data centers is challenging. Especially, the measurements and modeling of VDC requests on top of a physical data center can hardly been found in the recent literature as data traffic of network and cloud operators is often inaccessible to the outside world. However, according to recent research work, there is a consensus that the arrival rate of VDC requests follows the Poisson distribution; the lifetime of a VDC is exponentially distributed [6], [7], [8], [9]. In this work we also follow these configurations for VDC requests. Section V-A will discuss more about our test scenarios.

IV. ENERGY-AWARE VIRTUAL DATA CENTER

A. Energy-aware VDC architecture

In this work, we construct an energy-aware virtual data center architecture *EA-VDC* that allows deploying any energy-aware embedding algorithm. As shown in Fig. 1, the architecture consists of two main blocks: *Management* and

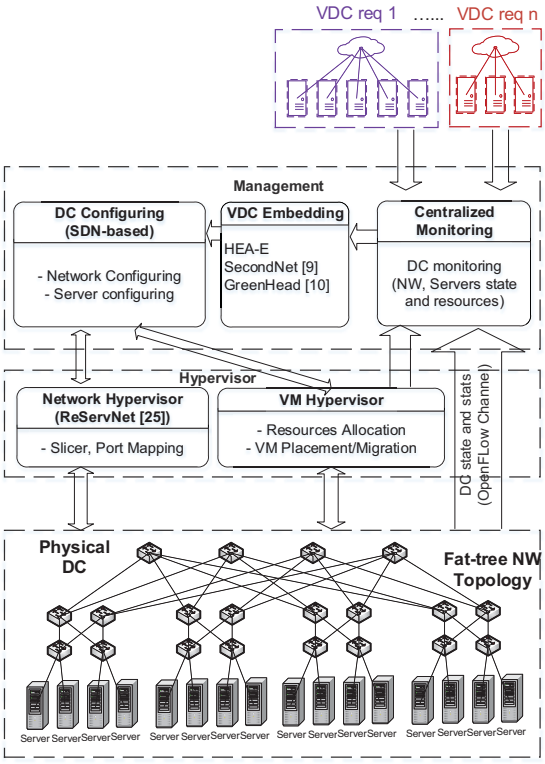

Fig. 1. Energy-Aware VDC Architecture

DC Hypervisor. The *Management* block consists of three subblocks: (1) *Centralized Monitoring* that monitors the states and resources of network devices and servers; (2) *VDC Embedding* that realizes various mapping algorithms; and (3) *DC Configuring* that controls the physical DC based on SDN technology by interacting with both the *Network Hypervisor* and *VM Hypervisor* subblocks. There are already several platforms for server hypervisor such as *HyperV, OpenStack, KVM, XEN, VMware* [26], [27]. We find that one of the most challenges of the VDC architecture is network hypervisor. In [22] we develop the ReServNet platform as an extension of FlowVisor [21] that deploys a new resource management and allocation concept for network virtualization. By combining server and network hypervisor based on SDN technology, the proposed architecture could be a promising approach for energy efficiency of DC virtualization.

B. Virtual Embedding Algorithm

This section describes our proposed *Heuristic Energy-Aware VDC Embedding* (HEA-E) algorithm for virtual data center embedding. The main objective of *HEA-E* is to reduce power consumption of the physical data center while improving the embedding acceptance rate. To the best our knowledge, there are two existing articles, SecondNet [11] and GreenHead [8], that focus on a sequence the VDC requests joins and leaves over time. Thus in this work, we compare our proposed

algorithm with the aforementioned approaches. There are some drawbacks of existing embedding algorithms. First, when embedding a VM request, the VM embedding algorithm in SecondNet [11] chooses a group of physical servers with the number of servers greater than or equal to the number of VMs. However, SecondNet does not take into account the availability of these servers as well as their available capacity in terms of memory and CPU. This directly affects the acceptance rate and energy saving level of the DC. Furthermore, the virtual link embedding algorithm performed afterwards uses the *Breadth First Search* (BFS) algorithm [25] to find the shortest path for link mapping. In some cases, the shortest path may require to turn on more immediate physical switches [28] instead of using ON_State switches that leads to more power consumption. On the other hand, GreenHead [8] makes use of $VLiM$ with BFS. Based on the BFS algorithm, GreenHead does focus on the available bandwidth of the links on shortest paths. If the available resources of the shortest path cannot satisfy the bandwidth demand, the VDC request is rejected.

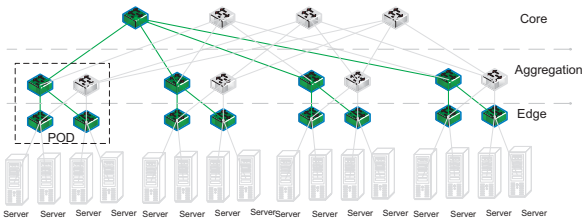

Fig. 2. Fat-tree with Minimum Spanning Tree

In contrast to GreenHead and SecondNet, HEA-E is an energy-efficient embedding algorithm that is based on two *Power Scaling* and *Idle Logic* approaches that have been attracted much attention from research communities [12]. *Power scaling* reduces power consumption of devices by adaptively changing operating rates of processing engines or link's speeds while *Idle logic* saves energy by quickly turning the devices off when there is no traffic load and rapidly waking these up if traffic is available. On the other hand, energy saving approaches should also maintain QoS and network reliability. That is, in case of no traffic the DC network maintains a *Minimum Spanning Tree* (MST) for minimum connectivity between servers. As traffic load increases, more servers and links in the Fat-Tree are turned on on demand [18]. In HEA-E, at initial phase when there is no traffic demand among servers, then: (1) all servers are turned off; (2) only one first left core switch Sw_{core} runs in lowest operating speed; (3) one first left aggregation switch Sw_{agg} and all access switches Sw_{acc} run in lowest operating speed. In this case there are only $\frac{k^2}{2}+k+1$ switches that are in ON_state (green switches in Fig.2). As a result, $\frac{5k^2}{4} - (\frac{k^2}{2} + k + 1)$ are in OFF_state.

The proposed embedding consists of *VmM* and *VLiM* that are described in Algorithm 1 and 2, respectively. When a VDC request arrives, HEA-E first uses *VmM* to map virtual machines onto physical servers. Subsequently, based on

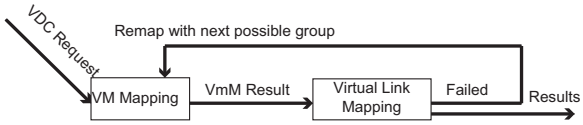

Fig. 3. VDC Embedding Flowchart

Algorithm 1 Virtual Machines Mapping algorithm

1: **input**: $G^p(t), R_j^v$
2: //Get list group in near, middle, far order
3: $GR \leftarrow getListGroup()$
4: $isSuccess =$ **False**
5: **for all** $gr_i \in GR$ **do**
6: $count = 0$
7: **for all** $S_j^v \in gr_i$ **do**
8: **if** $M_{cap}(S_j^v) > 0 \quad and \quad C_{cap}(S_j^v) > 0$ **then**
9: $count = count + 1$
10: **end if**
11: **end for**
12: **if** $count \geq |M^v|$ **then**
13: $list_{PossibleGroup} \leftarrow gr_i$
14: **end if**
15: **end for**
16: //Sort listPossibleGroup in increasing order by number of server need to turn on
17: *listSatisfiedGroup= tryMapAndSort(listPossibleGroup)*
18: *selectedGroup = getFirstGroup(listSatisfiedGroup)*
19: **if** *selectedGroup* $\neq \emptyset$ **then**
20: *isSuccess* = **True**
21: *vmResults* $\leftarrow$ *mapVM(*VM_j*, selectedGroup)*
22: **end if**
23: **output**: *isSuccess, vmResults*, G^p

VmM results, *VLiM* creates virtual links interconnecting newly mapped VMs on top of the physical network substrate. In order to improve the acceptance rate, in HEA-E we construct a re-mapping mechanism that allows re-mapping VmM if VLiM does not perform successfully. The flowchart of HEA-E is described in Fig. 3.

1) VmM: In this paper, we define three groups of servers, namely *near group, middle group* and *far group* that correspond to near traffic, middle traffic and far traffic, respectively (Line 2 of Alg. 1 – see also Sec. II-B). When a VDC request arrives, the following actions are taken:

- Step 1: Finding all possible mapping groups that have the number of available servers greater or equal to the number of VMs in the request. To improve reliability, a source and destination VM pair will not be mapped on the same physical server.
- Step 2: For all selected server groups, choose candidate groups with the least servers so that the power consumption of servers can be saved.
- Step 3: For all candidate groups with the least number of servers in Step 2, choose a group that has as many servers

that are near to each other as possible. That is, the candidate groups are prioritized in the $near \rightarrow middle \rightarrow far$ order. By doing this, it is likely that the virtual links interconnecting these VMs should traversed over less immediate hops, thus reducing the power consumption of network devices.

Algorithm 2 Virtual Link Mapping algorithm

input: $G^p(t), R_j^v$, *vmResults*
isSuccess = **False**
$L_j^v = sort(L_j^v$, *key= bwRequest, order= desc)*
for all *vLink* $\in L_j^v$ **do**
  $link_{type}$ = *getType(vmResults, vLink)*
  if $link_{type} \equiv$ *Near* **then**
    $sLink = \{sPhy, sEdge, dPhy\}$
    if *sLink = satisfied* **then**
      vLinkResults $\leftarrow$ *map(link, sLink)*
    end if
  else
    listSwitch = getListSwitch(*vmResults, vLink*)
    //Construct link from listSwitch
    listLink= constructLink(*listSwitch*)
    listLink = sort(*listSwitch, key= cap ,order=asc*)
    for all *sLink* $\in$ *listLink* **do**
      if *sLink = satisfied* **then**
        vLinkResults $\leftarrow$ *map(link, sLink)*
      end if
    end for
  end if
end for
if $|vLinkResults| = \left|L_j^v\right|$ **then**
  $isSucess = $ **True**
else
  Return to VM Mapping with next group
end if
output: $isSuccess$, $vLinkResults$, G^p

2) VLiM: The proposed virtual link mapping algorithm is inspired by Heller et al. [17] approach. Heller proposed the *Elastic Tree* concept in order to reduce consumed energy of DC network by maintaining a minimal logical topology on top of the Fat-Tree based on actual traffic demands. In HEA-E, VLiM (Algorithm 2) firstly arranges the matrix of virtual link requests of a VDC in non-increasing order of bandwidth demands. Then with each request corresponding to a source virtual machine vm_s and destination machine vm_d, the VLiM algorithm identifies the type of traffic scenario (i.e., *near, middle* or *far*) according to their relative positions that are already decided by the VmM algorithm. It then performs the correlative actions:

- *Near traffic* scenario: If the VMs are in the physical machines that are connected to the same edge switch sw_{edge}, the request is mapped onto the physical links connecting the VMs through that switch.
- *Middle* or *far traffic* scenarios: Otherwise, find all possible paths with necessary set of active switches. Select the path satisfied the traffic demand that has least available bandwidth and consumes the least additional energy.

If the virtual link mapping does not succeed, HEA-E performs virtual node re-mapping as described in Fig.3.

V. Performance Evaluation

To investigate the performance of the proposed algorithm HEA-E and the feasibility of the EA-VDC architecture, simulations have been carried on. The performance of HEA-E is compared with two existing VDC embedding algorithms, namely GreenHead [8] and SecondNet [11]. Three metrics have been used for the performance evaluation, which are *embedding acceptance ratio*, *power consumption*, and *complexity* in terms of running time.

A. Simulation environment

We developed a Java-based simulator for data center that makes use of k fat-tree topology. In this research $k = 4$ corresponds to a data center with 16 servers. The topology consists of the same $\frac{5k^2}{4}$*k-port* switches and $\frac{k^3}{4}$ servers. The power profile of switches and devices are taken from [14] and [15] (see Sec. III-B). Each server has 8 CPUs and $64GB$ of memory.

In the tests, similarly to previous work published in [6], [7], [8], [9], VDC requests are generated randomly following a Poisson distribution with arrival rate of $\lambda = 8$ VDCs per hour. The duration of a VDC is exponentially distributed with 2 hours average lifetime. In order to test the performance of embedding algorithms under different load scenarios, the number of VMs per VDC request varies from 4 to 16. All traffic demands between VM pairs are randomly distributed between $10Mbps$ and $90Mbps$, so that the data center utilization varies from 10% to 90%. The VMs within a VDC are interconnected with a random topology generated by Waxman algorithm [19]. Eq. 12 represents the probability that there exists a link connecting two arbitrary nodes u and v in the Waxman algorithm:

$$P(u, v) = \alpha e^{-d(u,v)/(\beta L)} \tag{12}$$

Parameters α, β in Eq.12 are in range of $(0, 1)$, d is the distance in Cartesian coordinates among VM u and v, L is the maximum distance between any two nodes in the graph. A rise in the parameter α increases the probability of existing links between any nodes in the graph, and an increase in β yields a larger ratio of long links to short links. Similar to some previous work [23], [24], in the simulations we set $\alpha = \beta = 0.5$ for average connectivity and link distance.

B. Experimental Results

1) Acceptance rate: As can be seen in Fig.4, HEA-E outperforms GreenHead and SecondNet in terms of acceptance ratio. Especially in highly loaded scenarios, the probability of successful VDC mapping in HEA-E is remarkably higher than the other two. The reason is, while taking into account the available resources of the physical substrate, HEA-E also has a flexible re-mapping mechanism.

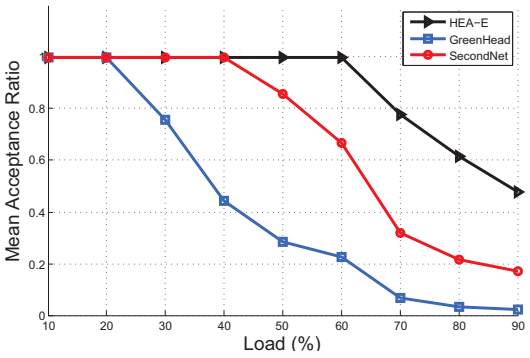

Fig. 4. Acceptance Rate

2) Resource-Efficiency Ratio: We define the *Resource Efficiency Ratio* (ReR) as the ratio of the number of accepted VDC requests over total available virtual machines in the physical DC. It is expected that with the same arrival rates of VDC requests λ, the physical DC resource usage dedicated to accepted VDCs be as high as possible. That is, with the same physical resources and arrival rate λ, the more ReR is, the more VDC requests can be accommodated.

In our experiments, a fat-tree with $k = 8$ or 128 servers is deployed. We assume that each physical server can accommodate 4VMs so that there are maximum 512VMs in a DC. The arrival rates of VDC requests vary from 10 to 80 requests per hour, each VDC request consists of 8VMs. As can be seen in Fig. 5, the Resource Efficiency Ratio of HEA-E is higher than those of SecondNet and GreenHead.

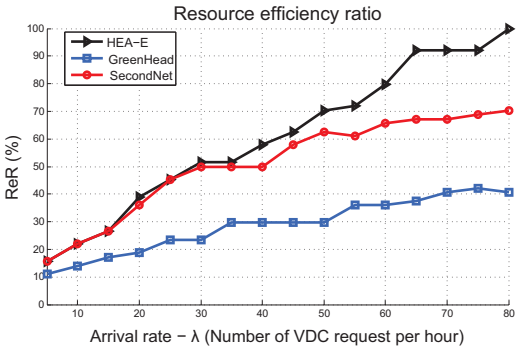

Fig. 5. Resource-efficiency Ratio

3) Power consumption: HEA-E prioritizes ON_State devices, including both servers and switches, it also places the VMs as near to each other as possible thus reducing the number of active network devices. Simulation results in Fig. 6 show that the power consumption of the new algorithm is better than SecondNet and as good as GreenHead.

We also measure the power consumption of each successful VDC as described in Eq.13 where $Num_{VDC}(t)$ is the number of mapped VDC. As can be seen in Fig.7, when the arrival

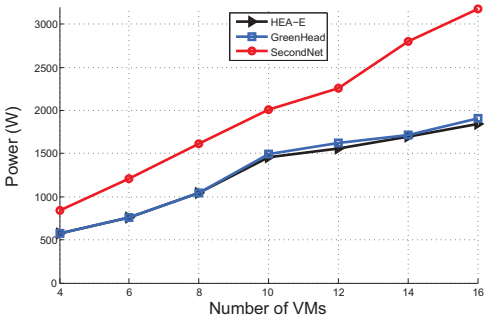

Fig. 6. Power Consumption

rates vary from 30 to 90 requests per hour, (1) the consumed power per each VDC is decreasing; (2) the consumed power of proposed algorithm HEA-E is less than both existing algorithms; and (3) as the requests rate increase, the power consumption of a VDC in case of HEA-E and GreenHead decreases very slowly, which implies that power consumption of the physical DC stays nearly linear to the number of embedded VDCs. We call this as the *power proportional* property.

$$Power_{VDC}(t) = \frac{P_N(t) + P_S(t)}{Num_{VDC}(t)} \tag{13}$$

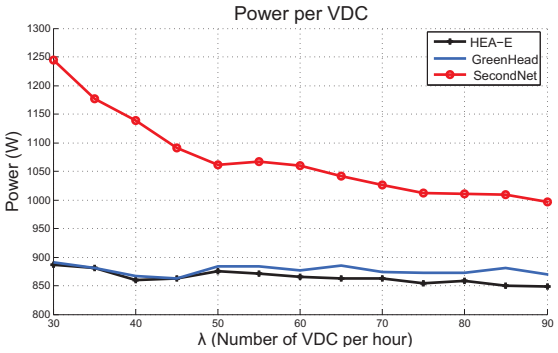

Fig. 7. Consumed power per VDC

In other scenario, arrival rate is 64 VDC requests per hour and each VDC request consists 8VMs. A $k = 8$ fat-tree and 128 servers is deployed (each server can accommodate 4VMs). The duration of VDC request is 2 hours. After $24h$, we measure the average consumed power of each VDC under three algorithms *HEA-E, GreenHead and SecondNet*. The experimental results are shown in Table III.

TABLE III
AVERAGE CONSUMED POWER PER VDC

	HEA-E	GreenHead	SecondNet
Consumed Power (W)	929.36	1080.30	1146.44

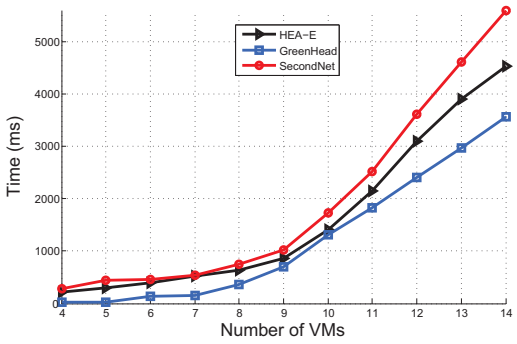

Fig. 8. Time Complexity

4) Time complexity: In order to evaluate the complexity of the embedding algorithms, we run different embedding requests on a *4GB Intel Core 2 Duo 2.67GHz* and measure the execution time. As shown in Fig.8, the execution time increases with the rise of VM requests. The execution time of GreenHead algorithm is the smallest. That is because after VmM, GreenHead uses BFS for *VLiM*, and if this shortest path does not satisfy the capability demand then this algorithm will drop the VDC request without any "re-mapping" process. Contrary to GreenHead, SecondNet and HEA-E have re-mapping technique so that it needs to take more time for calculation.

From the results there are some remarks as the follows: (1) Three energy-ware embedding algorithms are able to adapt the power consumption of the DC proportionally to the number of servers and the corresponding load. Among energy-aware embedding algorithms, HEA-E and GreenHead are the two best ones in terms of energy efficiency. Both algorithms can save upto 80% of the energy consumption of a 16-server fat-tree DC (i.e., $500W$ over total $3000W$); (2) However, the payoff of GreenHead is that its acceptance ratio is the worst. In contrast, HEA-E is the best algorithm in terms of acceptance ratio. That is, HEA-E can improve the acceptance ratio while maintaining its power consumption the lowest.

VI. Conclusion and Future Work

In this work we propose a virtual data center embedding algorithm as well as a new SDN-based virtualization architecture that allows deploying both network and server virtualization on the physical DC infrastructure. The newly proposed HEA-E embedding algorithm can reduce upto 80% of the data center energy consumption, while it can remarkably improve the acceptance ratio in comparison to some existing algorithms.

References

[1] P.Graubner, M.Schmidt, B.Freisleben, "Energy - efficient Virtual Machine Consolidation for Cloud Computing," IT Professional., pp. 114, 2012.
[2] F. Farahnakian, P. Liljeberg, J. Plosila, "Energy-Efficient Virtual Machines Consolidation in Cloud Data Centers Using Reinforcement Learning," 22nd Euromicro Int. Conf. Parallel, Distrib. Network-Based, 2014.
[3] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems," Advances in Computers, Vol. 82. 2011.
[4] PlannetLab project, http://www.planet-lab.org/. Accessed August 2014
[5] GENI, 2013, Global environment for network innovations. http://www.geni.net/. Accessed August 2015
[6] M. F. M. Zhani, Q. Zhang, G. Simon, and R. Boutaba, "VDC Planner : Dynamic Migration-Aware Virtual Data Center Embedding for Clouds," Integrated Network Management (IM 2013), pp. 18 25, 2013.
[7] Chowdhury, M., Member, S., Rahman, M. R., & Boutaba, R. ViNEYard : Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping. IEEE/ACM Transactions on Networking, 20(1), 206219(2012)
[8] A. Amokrane, M. F. Zhani, R. Langar, R. Boutaba, and G. Pujolle, "Greenhead: Virtual Data Center Embedding Across Distributed Infrastructures," IEEE Trans. Cloud Comput., vol. 1, pp. 36 49, Jan. 2013.
[9] Q. Zhang, M. F. Zhani, M. Jabri, and R. Boutaba, "Venice: Reliable virtual data center embedding in clouds," in IEEE INFOCOM 2014 - IEEE Conference on Computer Communications, 2014, pp. 289 297.
[10] Yoonseon Han, Jian Li, Jae-Yoon Chung, Jae-Hyoung Yoo, and J. W. Hong, "SAVE: Energy-aware Virtual Data Center embedding and Traffic Engineering using SDN," in Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft), 2015, no. Vm, pp. 1 9.
[11] C. Guo, G. Lu, H. J. 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. 6th ACM CONEXT 2010, pp. 15:1 15:12, 2010.
[12] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy Efficiency in the Future Internet: A Survey of Existing Approaches and Trends in Energy-Aware Fixed Network Infrastructures," Commun. Surv. Tutorials, IEEE, vol. 13, no. 2, pp. 223 244, 2011.
[13] A. Schrijver, "Theory of Linear and Integer Programming," John Wiley & Sons Inc, 1998.
[14] Priya Mahadevan, Puneet Sharma, Sujata Banerjee, Parthasarathy Ranganathan, "Energy Aware Network Operations," in Proceedings of INFOCOM, 2009.
[15] IBM Systems Magazine, IBM 2U rackmount 2 socket x86, Available on http://www.ibmsystemsmag.com/mainframe/Business-Strategy/ROI/energy_estimating/
[16] M. Al-Fares, A. Loukissas and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," ACM SIGCOMM, Aug. 2008
[17] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "ElasticTree : Saving Energy in Data Center Networks," NSDI 2010, pp. 1 17, 2010.
[18] H. T. Nguyen, N. N. Pham, T. H. Truong, N. T. Tran, M. D. Nguyen, V. G. Nguyen, T. H. Nguyen, Q. T. Ngo, D. Hock, and C. Schwartz, "Modeling and experimenting combined smart sleep and power scaling algorithms in energy-aware data center networks," Simul. Model. Pract. Theory, vol. 39, pp. 20 40, 2013.
[19] B.M. Waxman, "Routing of multipoint connections," IEEE J. Select. Areas Commun. 6 (9) (1988) 1617−1622
[20] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking : A Comprehensive Survey," Proc. IEEE, vol. 103, no. 1, pp. 14 76, 2015.
[21] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. Mckeown, and G. Parulkar, "FlowVisor: A Network Virtualization Layer," OpenFlow Switch Consortium, Tech. Rep., October 2009.
[22] H. T. Nguyen, A. V. Vu, D. L. Nguyen, V. H. Nguyen, M. N. Tran, Q. T. Ngo, T.-H. Truong, T. H. Nguyen, and T. Magedanz, "A generalized resource allocation framework in support of multi-layer virtual network embedding based on SDN," Comput. Networks, vol. 92, pp. 251 269, Dec. 2015.
[23] A. Fischer, J. F. Botero, M. Duelli, D. Schlosser, X. Hesselbach, and H. De Meer, "ALEVIN - a framework to develop, compare, and analyze virtual network embedding algorithms," in Electronic Communications of the EASST, vol. 37, pp. 112, 2011.
[24] D. Stezenbach, M. Hartmann, and K. Tutschku, "Parameters and challenges for virtual network embedding in the future Internet," in the First IEEE Workshop on Algorithms and Operating Procedures for Federated Virtualized Networks (FEDNET2012), 2012.
[25] S. Skiena, "The Algorithm Design Manual," Springer, 2008, p. 480.
[26] A. Desai, R. Oza, P. Sharma, and B. Patel, "Hypervisor : A Survey on Concepts and Taxonomy," no. 3, pp. 222 - 225, 2013.
[27] https://www.openstack.org/ Accessed October 2015
[28] J. F. Botero, X. Hesselbach, A. Fischer, and H. De Meer, "Optimal mapping of virtual networks with hidden hops," Telecommun. Syst., vol. 51, no. 4, pp. 273282, 2012.