SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

December 2016, Vol. 59 122307:1–122307:14
doi: 10.1007/s11432-016-5590-0

Virtual network embedding for power savings of servers and switches in elastic data center networks

Weigang HOU[1], Cunqian YU[1*], Lei GUO[1] & Xuetao WEI[2]

[1]*School of Computer Science and Engineering, Northeastern University, Shenyang* 110819*, China;*
[2]*School of Information Technology, University of Cincinnati, Cincinnati, OH* 2600*, USA*

Received November 13, 2015; accepted December 18, 2015; published online June 22, 2016

Abstract Currently, the elastic interconnection has realized the high-rate data transmission among data centers (DCs). Thus, the elastic data center network (EDCN) emerged. In EDCNs, it is essential to achieve the virtual network (VN) embedding, which includes two main components: VM (virtual machine) mapping and VL (virtual link) mapping. In VM mapping, we allocate appropriate servers to hold VMs. While for VL mapping, an optimal substrate path is determined for each virtual lightpath. For the VN embedding in EDCNs, the power efficiency is a significant concern, and some solutions were proposed through sleeping light-duty servers. However, the increasing communication traffic between VMs leads to a serious energy dissipation problem, since it also consumes a great amount of energy on switches even utilizing the energy-efficient optical transmission technique. In this paper, considering load balancing and power-efficient VN embedding, we formulate the problem and design a novel heuristic for EDCNs, with the objective to achieve the power savings of servers and switches. In our solution, VMs are mapped into a single DC or multiple DCs with the short distance between each other, and the servers in the same cluster or adjacent clusters are preferred to hold VMs. Such that, a large amount of servers and switches will become vacant and can go into sleep mode. Simulation results demonstrate that our method performs well in terms of power savings and load balancing. Compared with benchmarks, the improvement ratio of power efficiency is 5%–13%.

Keywords elastic data center network, green virtual network embedding, integrated optimization, load balancing, power efficiency

Citation Hou W G, Yu C Q, Guo L, et al. Virtual network embedding for power savings of servers and switches in elastic data center networks. Sci China Inf Sci, 2016, 59(12): 122307, doi: 10.1007/s11432-016-5590-0

1 Introduction

The number of data centers (DCs) supporting cloud computing and big data rapidly grows, and geographically dispersed DCs are interconnected by networking resources. Currently, the elastic interconnection has realized the high-rate data transmission among DCs [1] based on orthogonal frequency division multiplexing (OFDM) which provides fine-grained and orthogonal frequency slots of establishing elastic connections [2]. As a result, the elastic data center network (EDCN) emerged. An example of EDCN architecture can be seen in Figure 1.

* Corresponding author (email: yucunqian@163.com)

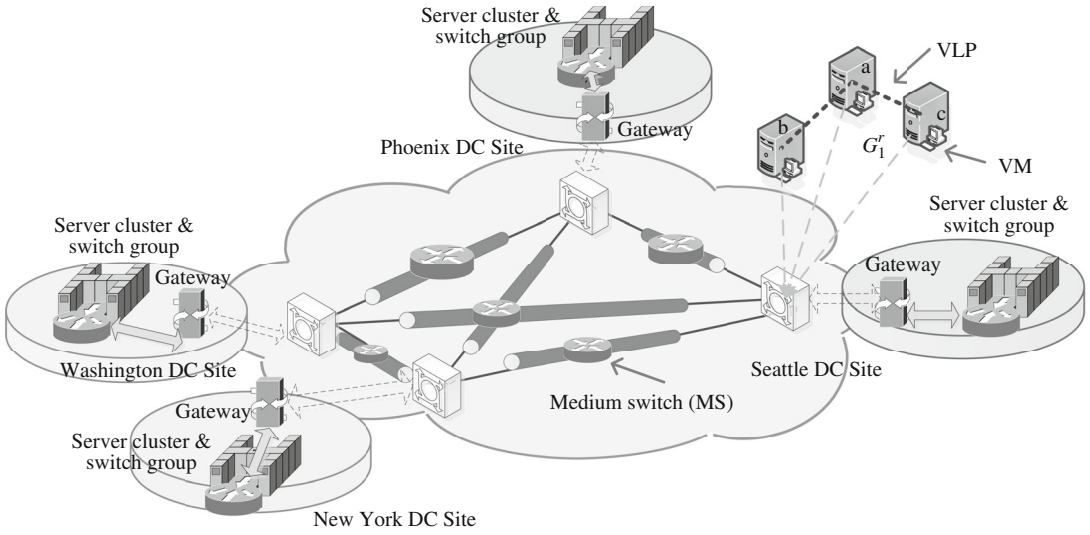

Figure 1 EDCN architecture.

Since the virtualization technology makes multi-tenant users able to share substrate computing and networking resources, it becomes imperative to achieve the virtual network (VN) embedding in EDCNs. A VN embedding request is denoted as a graph, where virtual machines (VMs) are interconnected with virtual links (VLs). Here, a VL is a virtual lightpath (VLP). This means that the VN embedding problem has two main components: VM mapping and VL mapping. In VM mapping, we try to allocate appropriate servers to hold VMs. While for VL mapping, an optimal substrate path should be determined for each VLP.

There is an increasing amount of application instantiations for VN embedding, such as large-scale cloud computing, 3D cloud rendering and currently-popular online shopping. These applications makes the power efficiency become a significant concern of the VN embedding in EDCNs. According to the potential increase of DCN traffic until 2016 [3], the majority of traffic is within DCs, since most applications hosted in DCs are based on parallel programming frameworks such as MapReduce [4]. The high interaction between distributed processing and storage servers is required to handle large sets of data (i.e., big data) [5]. Therefore, the traffic-dependent power consumption keeps on the trend of sharp growth with the increment of intra-DC communication flows [6, 7].

At present, some VN embedding solutions have been proposed to improve the intra-DC power efficiency, and they preferred to reduce the number of online servers through sleeping light-duty servers [8]. However, aside from servers, the switches in EDCNs are also power contributors. In fact, the interconnection of VMs is usually long-term and long-haul. Thus, the inter-VM traffic is the leading contributor of the energy consumption from switches along the substrate path. Since the medium switches (MSs) in Figure 1 (such as bandwidth-variable cross-connects) are responsible for the establishment of the long-term and long-haul interconnection among DCs, the power consumption proportional to time duration and transmission distance will become rather high. Additionally, the Top-of-Rack (ToR), aggregate switches (ASs) and core switches (CSs) consume around 10%–20% of the total power consumption in DC sites, and this is expected to further increase in the near future [9]. In a word, during the process of VN embedding in EDCNs, it is very necessary for us to save the power consumption of switches and servers.

Here is an example to illustrate the above viewpoint. As shown in Figure 2(a), to sleep more servers in DC_B, the traditional VN embedding solution tries to map VMs into DC_A, i.e., put VM b into DC_A, meanwhile, put VMs a and c into DC_B. However, as shown in Figure 2(a), the inter-DC traffic between VMs a and b or between VMs b and c actually consumes a great deal of power in the MSs along the substrate path between DC_A and DC_B. If we take the power saving of switches into account in VN embedding solutions, i.e., VMs a, b, and c are mapped together into DC_B in Figure 2(b), the power consumption of MSs will become negligibly small. Though scarifying a part of server power consumption,

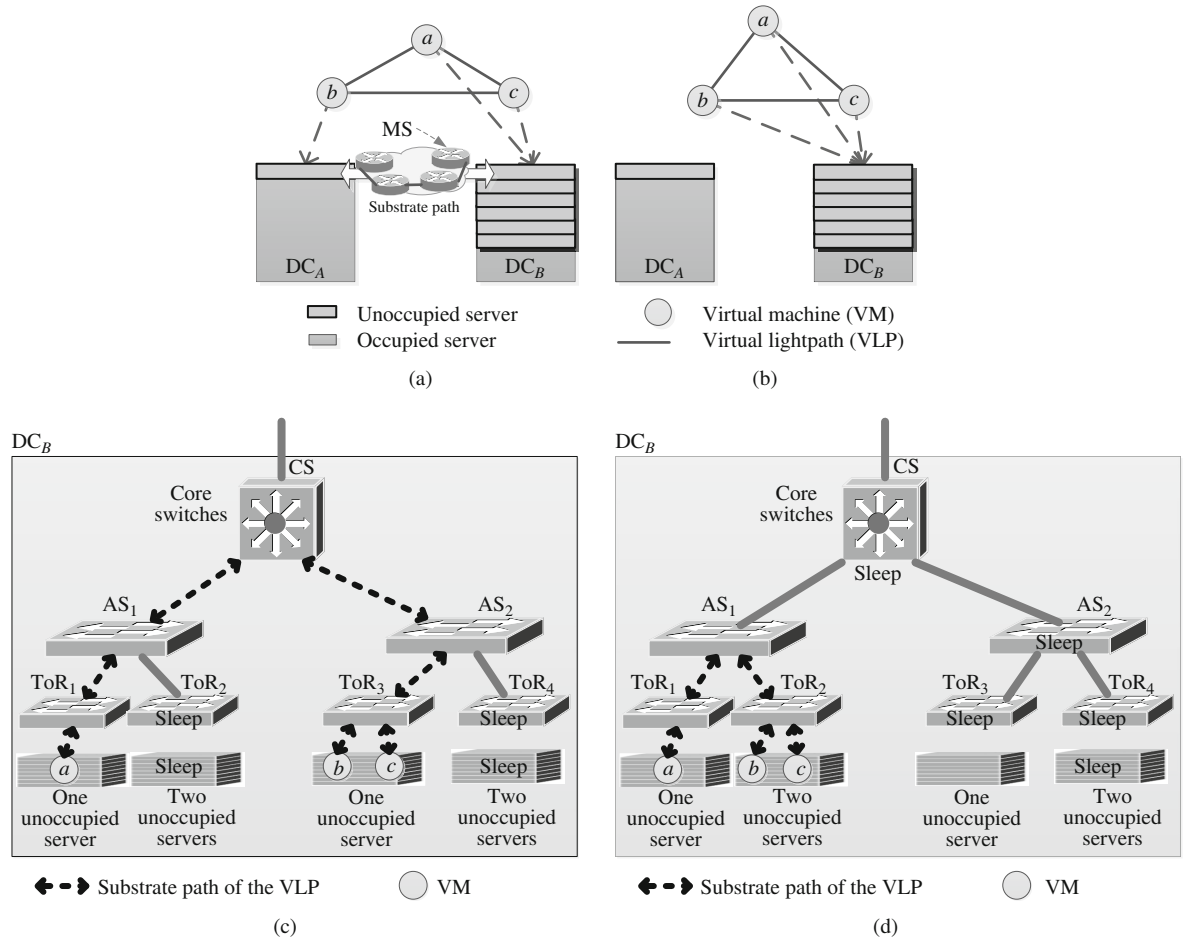

Figure 2 Comparison of existing and our VN embedding solutions. (a) Existing VN embedding solution; (b) our VN embedding solution; (c) existing VN embedding solution; (d) our VN embedding solution.

that will look very economical in a long term. After selecting the optimal DC, i.e., DC$_B$, the intra-DC mapping scheme becomes particularly important due to the existence of massive intra-DC traffic. As in Figure 2(c), DC$_B$ is typically organized in the hierarchical fat-tree architecture including totally 8 servers (two servers per ToR). The computing requirement of each VM is always smaller than a server capacity. Some previous intra-DC VN embedding solutions were achieved in Figure 2(c), where the substrate path of the VLP occupies one CS, two ASs (AS$_1$ and AS$_2$), and two ToRs (ToR$_1$ and ToR$_3$), so that more servers are unoccupied. While in our VN embedding solution, the servers with adequate computation capacity in the same cluster will be preferred to hold VMs a, b and c in Figure 2(d), which makes more switches go into sleep mode. Similarly, though scarifying a part of server power consumption, that also will look very economical in a long term. Finally, whatever power-saving technology is adopted, the DC-scale load balancing will be affected undoubtedly. A comprehensive consideration of load balancing and power-efficient VN embedding is pretty valuable.

To the best of our knowledge, this paper is the first design of VN embedding in EDCNs, with the objective of performing the power savings of servers and switches in a load-balancing manner. Our contributions are summarized as follows: (1) We mathematically formulated our problem with the consideration of the trade-off between power efficiency and load balancing in EDCNs. The NP-completeness was formally demonstrated. (2) A novel VN embedding heuristic was proposed by us, in order to achieve the power savings of servers and switches, and the improvement ratio can arrive up to 5%–14%.

The rest of this paper is organized as follows. Section 2 gives the problem formulation. Section 3 proposes our heuristic. We demonstrate simulation results in Section 4 before concluding this paper in Section 5.

Table 1 Parameters and variables

$P_{\text{ser}}^{\text{busy}}$: Working power of the server with the entire payload.
$P_{\text{ser}}^{\text{idle}}$: Power of the idle server.
$P_{\text{ToR}}^{\text{busy}}$: Working power of the ToR with with the entire payload.
$P_{\text{ToR}}^{\text{idle}}$: Power of the idle ToR.
$P_{\text{AS}}^{\text{busy}}$: Working power of the AS with with the entire payload.
$P_{\text{AS}}^{\text{idle}}$: Power of the idle AS.
$P_{\text{CS}}^{\text{busy}}$: Working power of the CS with with the entire payload.
$P_{\text{CS}}^{\text{idle}}$: Power of the idle CS.
$P_{\text{MS}}^{\text{busy}}$: Working power of the MS with with the entire payload.
$P_{\text{MS}}^{\text{idle}}$: Power of the idle MS.
U_{ToR}: The set of upstream (between ToR and AS) and downstream (between ToR and server) ports in the ToR.
U_{AS}: The set of upstream (between AS and CS) and downstream (between AS and ToR) ports in the AS.
U_{CS}: The set of upstream (between CS and gateway) and downstream (between CS and AS) ports in the CS.
U_{MS}: The set of MS ports.
$B_{\text{ss},\,j}^{\text{ToR}}$: The number of spectrum slots at the jth ($j \in U_{\text{ToR}}$) port in the ToR.
$B_{\text{ss},\,j}^{\text{AS}}$: The number of spectrum slots at the jth ($j \in U_{\text{AS}}$) port in the AS.
$B_{\text{ss},\,j}^{\text{CS}}$: The number of spectrum slots at the jth ($j \in U_{\text{CS}}$) port in the CS.
$B_{\text{ss},\,j}^{\text{MS}}$: The number of spectrum slots at the jth ($j \in U_{\text{MS}}$) port in the MS.
ξ_y^h: Boolean variable, if the VLP vl_y passes through the ToR ToR_h, $\xi_y^h = 1$; otherwise, $\xi_y^h = 0$.
ξ_y^i: Boolean variable, if the VLP vl_y passes through the AS AS_i, $\xi_y^i = 1$; otherwise, $\xi_y^i = 0$.
ξ_y^j: Boolean variable, if the VLP vl_y passes through the CS CS_j, $\xi_y^j = 1$; otherwise, $\xi_y^j = 0$.
ξ_y^k: Boolean variable, if the VLP vl_y passes through the MS MS_k, $\xi_y^k = 1$; otherwise, $\xi_y^k = 0$.
Rc_n: Positive integer variable, the amount of occupied computing resources in the server Ser_n.
δ_n: Utilization ratio of computing resources in the server Ser_n, and $0 \leqslant \delta_n < 1$.
$\delta_h^{m,r}$: Utilization ratio of spectrum slots in the ToR ToR_h, after serving the VN embedding request G_m^r, and $0 \leqslant \delta_h^{m,r} < 1$.
$\delta_i^{m,r}$: Utilization ratio of spectrum slots in the AS AS_i, after serving the VN embedding request G_m^r, and $0 \leqslant \delta_i^{m,r} < 1$.
$\delta_j^{m,r}$: Utilization ratio of spectrum slots in the CS CS_j, after serving the VN embedding request G_m^r, and $0 \leqslant \delta_j^{m,r} < 1$.
$\delta_k^{m,r}$: Utilization ratio of spectrum slots in the MS MS_k, after serving the VN embedding request G_m^r, and $0 \leqslant \delta_k^{m,r} < 1$.
$P_n(\delta_n)$: Power consumption of the server Ser_n.
P_{ser}: Power consumption of all servers.
$P_h^{m,r}$: Power consumption of the ToR ToR_h, after serving the VN embedding request G_m^r.
$P_i^{m,r}$: Power consumption of the AS AS_i, after serving the VN embedding request G_m^r.
$P_j^{m,r}$: Power consumption of the CS CS_j, after serving the VN embedding request G_m^r.
$P_k^{m,r}$: Power consumption of the MS MS_k, after serving the VN embedding request G_m^r.
P_{swt}: Power consumption of all switches.
$f_{u,v}^y$: Binary variable that equals to 1 if the traffic is on the connection link $e(u,v)$ of the VLP vl_y ($\text{vl}_y \in E_m^r$); otherwise, $f_{u,v}^y = 0$.
w_y: Positive integer variable, the index of the starting spectrum slot to be allocated to the VLP vl_y, and $1 \leqslant w_y \leqslant B$.
B: Maximal number of spectrum slots initially available per connection link.

2 Problem description

2.1 System model

The EDCN can be represented by a graph $G^{\text{DCN}}(\psi^{\text{DC}}, V^{\text{MS}}, E^M)$, where ψ^{DC} denotes the DC set, and $\psi^{\text{DC}} = \{G_\omega^{\text{DC}} | \omega \in (1, |\psi^{\text{DC}}|)\}$. Here, $|\psi^{\text{DC}}|$ is the total number of DCs; V^{MS} denotes the set of MSs, each of which is responsible for the establishment of inter-DC connections; E^M denotes the set of connection links between MSs.

Each DC is further represented by a graph $G_\omega^{\text{DC}}(V_\omega^s, E_\omega^s)$, where V_ω^s and E_ω^s represent the set of

substrate devices and connection links in this DC, respectively. In this paper, we take a hierarchical fat-tree DC architecture into account, then a DC has three levels of switches, i.e., CS, AS and ToR. Thus, we have $V_\omega^s = \{V_\omega^{\rm ser}, V_\omega^{\rm CS}, V_\omega^{\rm AS}, V_\omega^{\rm ToR}\}$, where $V_\omega^{\rm ser}$ denotes the set of servers in the DC $G_\omega^{\rm DC}$, $V_\omega^{\rm CS}$ denotes the CS set of $G_\omega^{\rm DC}$, $V_\omega^{\rm AS}$ denotes the AS set of $G_\omega^{\rm DC}$, and $V_\omega^{\rm ToR}$ denotes the ToR set of $G_\omega^{\rm DC}$. The server Ser_n has a certain amount of available computing capacity C_n. For each connection link, the entire spectrum is divided into a list of continuous spectrum slots, and all spectrum sots have the same granularity. Thus for the connection link e, the spectrum utilization can be recorded by a binary array b_e with B binary variables, where B represents the maximal number of spectrum slots initially available per connection link. More specifically, when the ith spectrum slot in the eth connection link is occupied, $b_e[i] = 1$, otherwise, $b_e[i] = 0$.

The set of VN embedding requests is $\psi^r = \{G_m^r | m \in (1, |\psi^r|)\}$. And the mth VN embedding request can be described by a graph $G_m^r(V_m^r, E_m^r)$, where the VM $\mathrm{vm}_x \in V_m^r$ has its computing requirement Rc_x, and the VLP $\mathrm{vl}_y \in E_m^r$ has the spectrum requirement Rs_y. We also list some important parameters and variables in Table 1.

2.2 Problem formulation

Based on the aforementioned system model and notation definitions, we first have the following objective function of power savings:

$$\text{Minimize}\ \ \mathrm{EE} = (P_{\rm ser} + P_{\rm swt}), \tag{1}$$

$$\text{s.t.}\ \ P_{\rm ser} = \sum_{\mathrm{Ser}_n \in \bigcup_\omega V_\omega^{\rm ser}} P_n(\delta_n), \tag{2}$$

$$P_n(\delta_n) = P_{\rm ser}^{\rm idle} + \left(P_{\rm ser}^{\rm busy} - P_{\rm ser}^{\rm idle}\right) \cdot \delta_n,\ \forall n, \tag{3}$$

$$\mathrm{RC}_n = \sum_{\mathrm{vm}_x \in \mathrm{Ser}_n} \mathrm{Rc}_x,\ \forall n, \tag{4}$$

$$\delta_n = \frac{\mathrm{RC}_n}{C_n},\ \ \forall n, \tag{5}$$

$$\begin{aligned} P_{\rm swt} = &\sum_{G_m^r \in \psi^r} \sum_{\mathrm{ToR}_h \in \bigcup_\omega V_\omega^{\rm ToR}} P_h^{m,r} + \sum_{G_m^r \in \psi^r} \sum_{\mathrm{AS}_i \in \bigcup_\omega V_\omega^{\rm AS}} P_i^{m,r} \\ &+ \sum_{G_m^r \in \psi^r} \sum_{\mathrm{CS}_j \in \bigcup_\omega V_\omega^{\rm CS}} P_j^{m,r} + \sum_{G_m^r \in \psi^r} \sum_{\mathrm{MS}_k \in V^{\rm MS}} P_k^{m,r} \end{aligned} \tag{6}$$

$$P_h^{m,r} = P_{\rm ToR}^{\rm idle} + \left(P_{\rm ToR}^{\rm busy} - P_{\rm ToR}^{\rm idle}\right) \cdot \delta_h^{m,r},\ \forall h,\ G_m^r, \tag{7}$$

$$\delta_h^{m,r} = \frac{\sum_{\mathrm{vl}_y \in E_m^r} \xi_y^h \cdot \mathrm{Rs}_y}{\sum_{j \in U_{\rm ToR}} B_{{\rm ss},j}^{\rm ToR}},\ \forall h,\ G_m^r, \tag{8}$$

$$P_i^{m,r} = P_{\rm AS}^{\rm idle} + \left(P_{\rm AS}^{\rm busy} - P_{\rm AS}^{\rm idle}\right) \cdot \delta_i^{m,r},\ \forall i,\ G_m^r, \tag{9}$$

$$\delta_i^{m,r} = \frac{\sum_{\mathrm{vl}_y \in E_m^r} \xi_y^i \cdot \mathrm{Rs}_y}{\sum_{j \in U_{\rm AS}} B_{{\rm ss},j}^{\rm AS}},\ \forall i,\ G_m^r, \tag{10}$$

$$P_j^{m,r} = P_{\rm CS}^{\rm idle} + \left(P_{\rm CS}^{\rm busy} - P_{\rm CS}^{\rm idle}\right) \cdot \delta_j^{m,r},\ \forall j,\ G_m^r, \tag{11}$$

$$\delta_j^{m,r} = \frac{\sum_{\mathrm{vl}_y \in E_m^r} \xi_y^j \cdot \mathrm{Rs}_y}{\sum_{j \in U_{\rm CS}} B_{{\rm ss},j}^{\rm CS}},\ \forall j,\ G_m^r, \tag{12}$$

$$P_k^{m,r} = P_{\rm MS}^{\rm idle} + \left(P_{\rm MS}^{\rm busy} - P_{\rm MS}^{\rm idle}\right) \cdot \delta_k^{m,r},\ \forall k,\ G_m^r, \tag{13}$$

$$\delta_k^{m,r} = \frac{\sum_{\mathrm{vl}_y \in E_m^r} \xi_y^k \cdot \mathrm{Rs}_y}{\sum_{j \in U_{\rm MS}} B_{{\rm ss},j}^{\rm MS}},\ \forall k,\ G_m^r. \tag{14}$$

As shown in (1), we try to minimize the power consumption of servers and switches in the EDCN. Eqs. (2)–(5) calculate the total power consumption of all servers, while Eqs. (6)–(14) calculate the total power consumption of switches with different levels. Here, we utilize a velocity model, i.e., the power consumption of servers and switches is determined according to the utilization ratio of computing and spectral resources. In addition, as shown in (8), the lower part of this equation records the total number of available spectrum slots initially assigned for upstream and downstream ports in ToR_h. Since the upstream port supports the traffic routing between ToR and AS, it initially has the less available spectrum slots than that of the downstream port supporting the traffic routing between ToR and server. AS and CS both have the similar setting of spectrum slots with ToR.

To achieve load balancing, we have the following objective function:

$$\begin{aligned}\text{Minimize LB} = &\sum_{\mathrm{Ser}_n \in \bigcup_\omega V_\omega^{\mathrm{ser}}} \left[1 - \sum_{\mathrm{vm}_x \in \mathrm{Ser}_n} \frac{(\xi_n^x \cdot \mathrm{Rc}_x)}{C_n}\right] \\ &+ \sum_{\mathrm{vl}_y \in E_1^r \cup E_2^r \cup \cdots \cup E_{|\psi^r|}^r} \left[w_y + \sum_{(u,v) \in \bigcup_\omega E_\omega^s} (f_{u,v}^y \cdot \mathrm{Rs}_y)\right]. \end{aligned} \tag{15}$$

To ensure the load balancing of computing resources, we consolidate evenly VMs into more servers, which is demonstrated in the left part of (15) that has the minimum value when $\sum_{\mathrm{vm}_x \in \mathrm{Ser}_n} (\xi_n^x \cdot \mathrm{Rc}_x)$ is equal to C_n (i.e., 100% utilization ratio for every server). Similarly, to save spectral resources, we minimize the maximal index of the occupied spectrum slot in the EDCN, as illustrated by the right part of (15).

Joint Optimization: Due to the contradiction between the optimization objectives of minimizing EE and LB, we have the following joint optimization objective.

Definition 1. Let $\mathrm{LB}_{\mathrm{best}}$ and $\mathrm{LB}_{\mathrm{worst}}$ denote the optimal and the worst solutions of (15), respectively. Let $\mathrm{EE}_{\mathrm{best}}$ and $\mathrm{EE}_{\mathrm{worst}}$ denote the optimal and the worst solutions of (1), respectively. s_{LB} is the current solution of (15), and s_{EE} is the current solution of (1). If and only if the following equation is satisfied, the solution is fair.

$$\frac{\mathrm{LB}_{\mathrm{worst}} - s_{\mathrm{LB}}}{\mathrm{LB}_{\mathrm{worst}} - \mathrm{LB}_{\mathrm{best}}} = \frac{\mathrm{EE}_{\mathrm{worst}} - s_{\mathrm{EE}}}{\mathrm{EE}_{\mathrm{worst}} - \mathrm{EE}_{\mathrm{best}}}. \tag{16}$$

Theorem 1. Joint optimization problem is non-linear and non-convex.

Proof. For the VLPs owned by a VN embedding request, they should consume the same spectral range, in order to satisfy the constraints of spectrum continuity and non-overlapping[1)]. Since these constraints are non-linear, our problem is non-linear. Meanwhile, Eq. (16) makes the optimization problem non-convex, i.e., it is very hard to get the optimal solution in polynomial time. Therefore, we propose the novel heuristic adaptive to network planning and dynamic online phases.

3 Heuristic

In this section, we design optimizing schemes for the heuristic called integrated green VN embedding (IGVE), in order to minimize the power consumption of servers and switches in EDCNs. In our heuristic, we insert all VN embedding requests into the queue Q^r in ascending order according to their arriving time, and then we serve them one-by-one starting from the top of Q^r, until all requests are processed. For each VN embedding request, three schemes are utilized, mainly including the selection of target DC(s), VM mapping, and VL mapping. After performing VM and VL mapping processes, we update the resource utilization of the substrate EDCN, and calculate the corresponding power consumption. The detailed pseudo code of our heuristic is shown in Algorithm 1. In the following, we describe three schemes in detail.

1) Non-overlapping constraint: if multiple VLPs go through the same connection link, their occupied spectrum slots cannot be overlapped.

Algorithm 1 IGVE

Input: DC set ψ^{DC}, and VN embedding request set ψ^r.
Output: VM-mapping result MAP_N, VL-mapping result MAP_L and the total power consumption s_{EE}.

Request queue $Q^r \leftarrow \{\psi^r, \text{arriving time}, '\text{ ascending}'\}$;
Calculate residual resource $R^{\mathrm{DC}}_{\omega,\mathrm{avl}}$ for each DC $G^{\mathrm{DC}}_{\omega} \in \psi^{\mathrm{DC}}$, according to (17);
Candidate DC queue $Q^{\mathrm{DC}} \leftarrow \{\psi^{\mathrm{DC}}, R^{\mathrm{DC}}_{\omega,\mathrm{avl}}, '\text{ descending}'\}$;
Candidate multi-DC cluster queue $Q^{\mathrm{multi}}_{\mathrm{DC}} \leftarrow \{(G^{\mathrm{DC}}_{\omega_{i_1}}, G^{\mathrm{DC}}_{\omega_{j_1}}), \ldots, (G^{\mathrm{DC}}_{\omega_{ic}}, G^{\mathrm{DC}}_{\omega_{jc}}), ' L'_{\omega_i,\omega_j}, ' \text{ ascending}'\}$;
while $Q^r \neq \varphi$ **do**
  $G^r_m = Q^r.top()$;
  while $(Q^{\mathrm{DC}} \cup Q^{\mathrm{multi}}_{\mathrm{DC}}) \neq \emptyset$ **do**
    Target DC $G^{\mathrm{DC}}_{\omega*} = Q^{\mathrm{DC}}.top()$;
    if $Q^{\mathrm{DC}}.empty() == 1$ **then**
      Form a new DC cluster $G^{\mathrm{DC}}_{\omega*} = Q^{\text{multi-DC}}.top()$;
    end if
    $\rho(\mathrm{MAP}_N, \mathrm{MAP}_L, P_{\mathrm{vl}_y}) = \mathrm{MAP}(G^r_m, G^{\mathrm{DC}}_{\omega*})$;
    if $\rho(\mathrm{MAP}_N, \mathrm{MAP}_L, P_{\mathrm{vl}_y}) == 1$ **then**
      $P_{\mathrm{swt}} += P_{\mathrm{vl}_y}$;
      break;
    else if $\rho(\mathrm{MAP}_N, \mathrm{MAP}_L, P_{\mathrm{vl}_y}) == 0$ & $Q^{\mathrm{DC}}.empty()! = 1$ **then**
      $Q^{\mathrm{DC}}.pop()$;
      continue;
    else
      $Q^{\mathrm{multi}}_{\mathrm{DC}}.pop()$;
      continue;
    end if
  end while
  if embedding succeed **then**
    Update $G^{\mathrm{DC}}_{\omega*}$ accordingly;
    Output MAP_N and MAP_L;
  else
    Mark G^r_m as blocked;
  end if
  $Q^r.pop()$;
end while
Calculate the power consumption of servers P_{ser} according to (2);
$s_{\mathrm{EE}} \leftarrow P_{\mathrm{swt}} + P_{\mathrm{ser}}$;
Output s_{EE};

3.1 Selection of target DC(s)

We let Q^{DC} denote the queue of candidate DCs, and we put all candidate DCs into Q^{DC} in descending order according to $R^{\mathrm{DC}}_{\omega,\mathrm{avl}}$ that denotes the residual available computing and spectral resources of the ωth DC. The value of $R^{\mathrm{DC}}_{\omega,\mathrm{avl}}$ can be computed by using the following equation.

$$R^{\mathrm{DC}}_{\omega,\mathrm{avl}} = \frac{C^{\mathrm{DC}}_{\omega,\mathrm{avl}}}{C^{\mathrm{DC}}_{\omega}} + \frac{B^{\mathrm{DC}}_{\omega,\mathrm{avl}}}{B^{\mathrm{DC}}_{\omega}}, \tag{17}$$

where $C^{\mathrm{DC}}_{\omega,\mathrm{avl}}$ and $B^{\mathrm{DC}}_{\omega,\mathrm{avl}}$ represent the residual available computing and spectral resources, respectively. C^{DC}_{ω} and B^{DC}_{ω} are initial computing capacity and spectral resource of the ωth DC, respectively. Since the magnitude orders of computing and spectral resources are different, we utilize the ratios in (17) to eliminate a possible magnitude gap.

As discussed in the example of Figure 2(a), to save the power consumption of switches, we tend to select a single DC to hold as many VMs as possible, i.e., the top element of Q^{DC} is the preferred target DC for the current VN embedding request. Certainly, multiple DCs have to be combined into a group to hold VMs, when a single DC has insufficient resources. The DCs in the same group can belong to one service provider or organization. Let $Q^{\mathrm{multi}}_{\mathrm{DC}}$ denote the queue of candidate DC groups. Each group has two DCs, and the distance between each other is recorded as L_{ω_i,ω_j}. Put all candidate DC groups into $Q^{\mathrm{multi}}_{\mathrm{DC}}$ in ascending order according to the distance L_{ω_i,ω_j}. Thus, we have $Q^{\mathrm{multi}}_{\mathrm{DC}} = \{(G^{\mathrm{DC}}_{\omega_{i_1}}, G^{\mathrm{DC}}_{\omega_{j_1}}), (G^{\mathrm{DC}}_{\omega_{i_2}}, G^{\mathrm{DC}}_{\omega_{j_2}}) \cdots (G^{\mathrm{DC}}_{\omega_{ic}}, G^{\mathrm{DC}}_{\omega_{jc}})\}$, $L_{\omega_{i_1},\omega_{j_1}} \leqslant L_{\omega_{i_2},\omega_{j_2}} \leqslant \cdots \leqslant L_{\omega_{i_c},\omega_{j_c}}$ and $c = \lceil |\psi^{\mathrm{DC}}|/2 \rceil$. The top of $Q^{\mathrm{multi}}_{\mathrm{DC}}$

Algorithm 2 VM mapping

Input: VN embedding request G_m^r, and the target DC $G_{\omega*}^{\mathrm{DC}}$.
Output: MAP_N and MAP_L.

Remove resource-inadequate components and construct IAG;
for resource-adequate components in IAG **do**
  $\mathrm{SAG}_k \leftarrow$ select inter-connected components in IAG;
  if $\mathrm{NodeNum}(\mathrm{SAG}_k) \geqslant |V_m^r|$ **then**
    Record SAG_k;
  end if
end for
$\mathrm{SAGS} \leftarrow \{\mathrm{SAG}_k, \mathrm{NodeNum}(\mathrm{SAG}_k), '\,\mathrm{descending}'\}$;
while $\mathrm{SAGS} \neq \emptyset$ **do**
  $\mathrm{SAG}_k = \mathrm{SAGS}.top()$;
  $Q^{\mathrm{vm}} \leftarrow \{\mathrm{vm}_x \in V_m^r, \mathrm{NodeDegree}(\mathrm{vm}_x), '\,\mathrm{descending}'\}$;
  for $\forall \mathrm{Ser}_n \in \mathrm{SAG}_k$ **do**
    Calculate R_n^{ARR} according to (18);
  end for
  $Q^{\mathrm{server}} \leftarrow \{\mathrm{Ser}_n \in \mathrm{SAG}_k, '\,R_n^{\mathrm{ARR}}, '\,\mathrm{descending}'\}$;
  $\mathrm{vm}_{x*} \leftarrow Q^{\mathrm{vm}}.top()$ and $\mathrm{Ser}_{n*} \leftarrow Q^{\mathrm{server}}.top()$;
  $\mathrm{MAP}_N(\mathrm{vm}_{x*}) = \mathrm{Ser}_{n*}$, $C_{n*} \leftarrow C_{n*} - \mathrm{Rc}_{x*}$;
  $Q^{\mathrm{server}}.pop()$ and $Q^{\mathrm{vm}}.pop()$;
  $Q^{\mathrm{server}} \leftarrow \{\mathrm{Ser}_n \in Q^{\mathrm{server}}, '\,\mathrm{Ser}_{n*}.\mathrm{intra-cluster\_list}'\}$;
  while $Q^{\mathrm{vm}} \neq \emptyset$ **do**
    $\mathrm{vm}_{x*} \leftarrow Q^{\mathrm{vm}}.top()$, $\mathrm{Ser}_{n*} \leftarrow Q^{\mathrm{server}}.top()$, $\mathrm{MAP}_N(\mathrm{vm}_{x*}) = \mathrm{Ser}_{n*}$;
    $C_{n*} \leftarrow C_{n*} - \mathrm{Rc}_{x*}$, $Q^{\mathrm{server}}.pop()$ and $Q^{\mathrm{vm}}.pop()$;
  end while
  for $\mathrm{vl}_y \in E_m^r$ **do**
    Find the substrate path P_{opt} in SAG_k, assign contiguous spectrum slots for P_{opt};
    $\mathrm{MAP}_L(\mathrm{vl}_y) = P_{\mathrm{opt}}$, and calculate P_{vl_y};
    Update the link cost of $e \in P_{\mathrm{opt}}$;
  end for
  if embedding succeed **then**
    Output MAP_N, MAP_L and P_{vl_y};
  end if
end while

is the preferred target DC group.

3.2 VM mapping

During the process of VM mapping, the VMs of the current VN embedding request should be consolidated into appropriate servers in target DC(s). The detailed procedure of VM mapping is shown in Algorithm 2. In the target DC $G_{\omega^*}^{\mathrm{DC}}$, the corresponding integrated auxiliary graph (IAG) is constructed according to the spectral utilization of connection links and the residual available computing resource of servers in this target DC. More specifically, IAG consists of independent sub-auxiliary graphs (SAGs), and one group of resource-sufficient servers interconnected with available connection links will form a SAG. This is because that after resource-insufficient servers and connection links are deleted from the target DC, different groups of resource-sufficient servers (different SAGs) may be separated from each other. The SAGs in an IAG are sorted in descending order according to the number of servers in each SAG, and we first select the top one to perform VM mapping process.

For the VM mapping process based on the SAG_k, we have following operations.

Firstly, the VMs of the current VN embedding request are sorted in descending order according to their node degree. In other words, the VM with the largest node degree will be mapped in prior.

Now, the question is which server in SAG_k will hold the top VM? For the target DC $G_{\omega^*}^{\mathrm{DC}}$, we construct its intra-cluster server lists according to different switch levels, e.g., the servers connecting the same ToR_h form the server list of the ToR_h cluster, which is recorded as $V_{\mathrm{clus}}^{\mathrm{ToR}_h,\omega*}$, and the servers connecting the same AS_i form the server list of the AS_i cluster, which is recorded as $V_{\mathrm{clus}}^{\mathrm{AS}_i,\omega*}$. A series of intra-cluster server lists will be managed by an upper traffic controller. After constructing intra-cluster server lists,

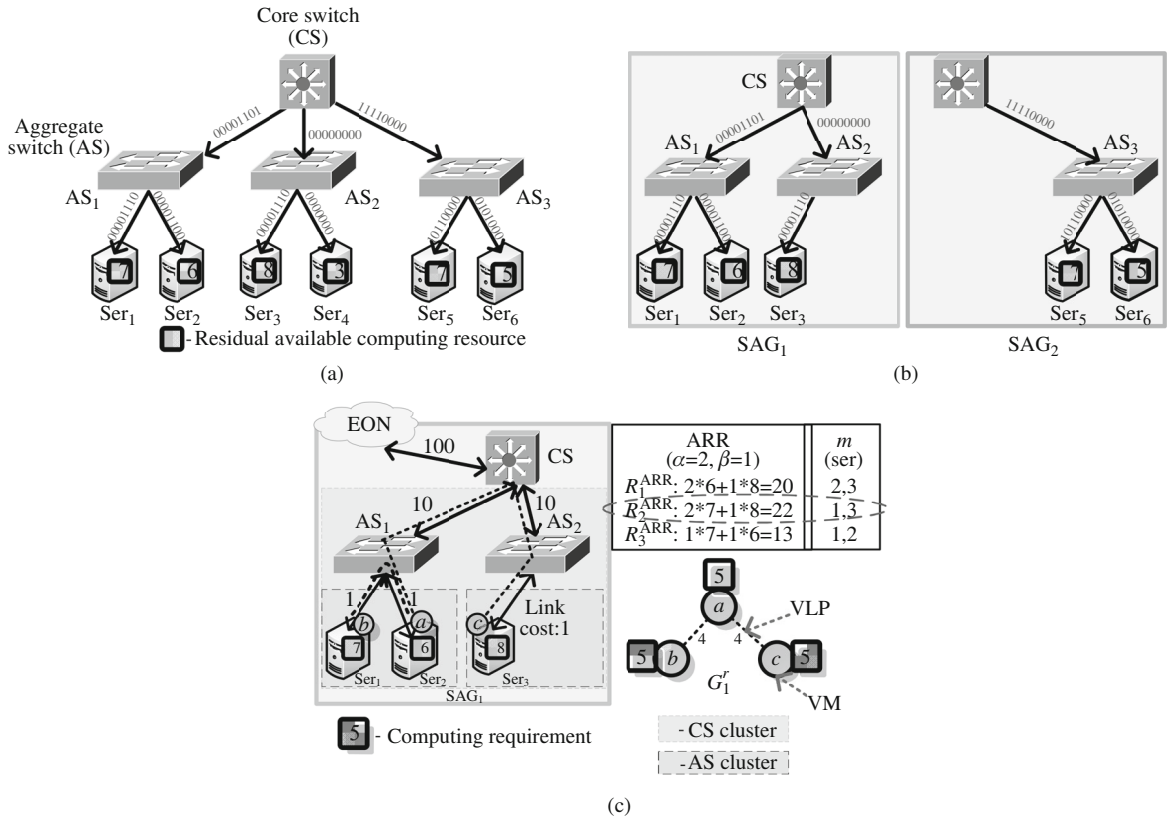

Figure 3 An illustration of IGVE. (a) Seattle DC: a fat-tree DC including 6 servers and 4 switches; (b) IAG including SAG_1 and SAG_2; (c) VM and VL mapping processes based on SAG_1.

we calculate the adjacent residual resource (ARR) information for each server in SAG_k by using the following equation.

$$R_n^{\mathrm{ARR}} = \alpha \cdot \sum_{\mathrm{Ser}_m \in V_{\mathrm{clus}}^{\mathrm{ToR}_h,\omega*}, \mathrm{Ser}_n \in V_{\mathrm{clus}}^{\mathrm{ToR}_h,\omega*}, n \neq m} \left[C_m - \sum_{\mathrm{vm}_x \in \mathrm{Ser}_m} \mathrm{Rc}_x \right] + \beta \cdot \sum_{\mathrm{Ser}_m \in (V_{\mathrm{clus}}^{\mathrm{AS}_i,\omega*} - V_{\mathrm{clus}}^{\mathrm{ToR}_h,\omega*}), \mathrm{Ser}_n \in V_{\mathrm{clus}}^{\mathrm{ToR}_h,\omega*}} \left[C_m - \sum_{\mathrm{vm}_x \in \mathrm{Ser}_m} \mathrm{Rc}_x \right] \forall \mathrm{Ser}_n \in V_{\omega*}^{\mathrm{ser}} \cap \mathrm{SAG}_k, \quad (18)$$

where α ($A \geqslant \alpha > 1$) and β ($0 < B \leqslant \beta \leqslant 1$) are both weight coefficients. The thresholds A and B are pre-determined according to Service Level Agreement (SLA). Correspondingly, the serves in the same ToR cluster (i.e., the first product item of (18)) has the higher priority to hold VMs, compared with the servers in the same AS cluster but different ToR clusters (i.e., the second product item of (18)). Then, we re-sort servers in descending order according to their ARR value. Naturally, the VM with the largest node degree will be mapped into the server with the biggest ARR value in SAG_k, since it facilitates the power saving of switches.

3.3 VL mapping

After executing the VM mapping, we continuingly perform the corresponding VL mapping on the same SAG. The connection links in the SAG have various kinds of costs, i.e., the upper switch-layer link takes the higher cost. The reason is that the upper-layer switch is regarded as a common root node and plays an important role on the aggregation of the traffic from lower-layer switches. Thus, as an example of Figure 3(c), the costs of the connection links on CS, AS and ToR layers are respectively 100, 10, and 1. Note that the CS layer is the topmost one, while AS and ToR layers correspond to CS and AS clusters,

respectively. According to the VM mapping result and link cost, the minimal-cost path is computed by using Dijkstra as the substrate path of the VLP in the current VN embedding request. The spectrum continuity and non-overlapping constraints should be satisfied.

3.4 IGVE Illustration

As shown in Figure 1, there are four DCs in the EDCN, mainly including Phoenix (G_1^{DC}), Washington (G_2^{DC}), New York (G_3^{DC}) and Seattle (G_4^{DC}) DC sites. The VN embedding request G_1^r currently arrives at this EDCN. The computing requirement of the VMs in G_1^r has 5 units of computing resources, i.e., $\mathrm{Rc}_x = 5$. And the spectrum requirement of the VLPs in G_1^r has four continuous spectrum slots, i.e., $\mathrm{Rs}_y = 4$. As shown in Figure 3(a), G_4^{DC} totally has 36 units of residual available computing resources, i.e., $C_{4,\mathrm{avl}}^{\mathrm{DC}} = 36$; G_4^{DC} totally has 52 available spectrum slots, i.e., $B_{4,\mathrm{avl}}^{\mathrm{DC}} = 52$; the initial computing capacity of G_4^{DC} totally has 54 units of computing resources because it has 6 servers, each of which initially has 9 units of available computing resources, i.e., $C_4^{\mathrm{DC}} = 54$; the initial spectral resource of G_4^{DC} totally has 72 spectrum slots because we have 9 connection links, each of which initially has 8 spectrum slots, i.e., $B_4^{\mathrm{DC}} = 72$. Thus, in Figure 3(a), the spectrum utilization of each connection link is recorded by the binary array with 8 binary variables (0: unoccupied, 1: occupied). Since $C_{4,\mathrm{avl}}^{\mathrm{DC}} = 36$, $B_{\omega,\mathrm{avl}}^{\mathrm{DC}} = 52$, $C_4^{\mathrm{DC}} = 54$, and $B_4^{\mathrm{DC}} = 72$, $R_{4,\mathrm{avl}}^{\mathrm{DC}} \approx 1.39$ according to (17). Let $R_{4,\mathrm{avl}}^{\mathrm{DC}}$ have the largest value in this illustration so that Seattle (G_4^{DC}) DC site can be selected as the target DC for embedding G_1^r.

Inside Seattle (G_4^{DC}) DC, the corresponding IAG is constructed in Figure 3(b). After deleting resource-insufficient servers (i.e., Ser_4 that merely has 3 units of residual computing resources in Figure 3(a) but the computing requirement of each VM in G_1^r is 5) from Seattle DC, the IAG has two independent SAGs. As shown in Figure 3(b), in SAG_1, the first four continuous spectrum slots on each connection link are available for the spectrum requirement of each VLP in G_1^r with satisfying spectrum continuity and non-overlapping constraints; in SAG_2, the last four continuous spectrum slots are available on each connection link. Since SAG_1 has more servers, we select it to perform the following VM mapping for G_1^r.

SAG_1 has the server lists of two AS clusters. As shown in Figure 3(c), the server list of AS_1 cluster includes two servers Ser_1 and Ser_2, while the server list of AS_2 cluster includes one server Ser_3. We then calculate the ARR value for each server in SAG_1, according to (18). Here, we let $\alpha = 2$ and $\beta = 1$, thus $R_1^{\mathrm{ARR}} = 20$, $R_2^{\mathrm{ARR}} = 22$, and $R_3^{\mathrm{ARR}} = 13$. Naturally, the VM with the largest node degree (i.e., VM a) is mapped into the server Ser_2 with the biggest ARR value in SAG_1. Similarly VMs b and c are mapped into servers Ser_1 and Ser_3, respectively.

Finally, we determine the substrate path in Figure 3(c) for the VLP of G_1^r, according to the link cost and the above VM mapping results. We can see that two substrate paths totally occupy 6 connection links, and only one of them traverses along the CS. While for the other VN embedding solutions, such as $\mathrm{vm}_a \in \mathrm{Ser}_3$, $\mathrm{vm}_b \in \mathrm{Ser}_1$, and $\mathrm{vm}_c \in \mathrm{Ser}_2$, both of two substrate paths will traverse along the CS. Above all, our method saves more power of switches.

3.5 Time complexity

The time complexity of IGVE mainly depends on the target DC selection and VM mapping. The time complexity of the target DC selection is $O(|\psi^{\mathrm{DC}}|^2)$, while the time complexity of the VM mapping is $O(M \log M + M)$, where M is the total number of VMs. In addition, the time complexity of route searching and VL mapping inside a target DC is $O[|V_\omega^s|^2(|E_\omega^s| + |V_\omega^s| \log_2 |V_\omega^s|)]$. And if VMs will be mapped into different DCs, the time complexity of inter-DC VL mapping is

$$O\left\{\left(\left|\psi^{\mathrm{DC}}\right| + \left|V^{\mathrm{MS}}\right|\right)^2 \left[\left|E^M\right| + \left(\left|\psi^{\mathrm{DC}}\right| + \left|V^{\mathrm{MS}}\right|\right) \log_2 \left(\left|\psi^{\mathrm{DC}}\right| + \left|V^{\mathrm{MS}}\right|\right)\right]\right\}.$$

4 Simulation results and discussions

In this section, we first introduce simulation settings, benchmarks and performance metrics. Then, we discuss the comparative results between our IGVE and benchmarks in terms of performance metrics.

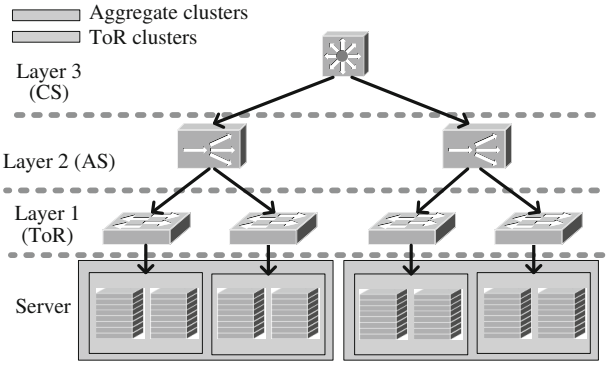

Figure 4 Simulation topology of DC.

Table 2 Power-consuming parameter settings

Devices	Power (W)	Devices	Power (W)
ToR:		CS:	
- Idle power	235	- Control logic (1×node) − Idle power	49638
- Working power	650	- Working power (16×ports)	50482
AS:		- SOA switch (1×port)	20
- Resource allocator (1×node) − Idle power	296	- MEMS switch (1×port)	0.1
- Working power (16×ports)	2808	- Tunable wavelength converter (1×port)	1.69
- Classifier (1×port)	62	- Control info extraction/re-insertion (1×port)	17
- Assembler (1×port)	62	- Optical amplifiers (1×port)	14
- Packet extractor (1×port)	25	Server:	
- Switch (1×port)	8	- Power consumption of an idle server	202
		- Power consumption of the server with full duty	255

4.1 Simulation setting

We take VC++ to build the testbed, and utilize the topology in Figure 1 as the simulation network. Each DC in the simulation network has a typical 3-layered structure as shown in Figure 4. We can see that each DC contains 1 CS, 2 ASs, 4 ToRs and 8 servers. The 1-layer connection link contains 128 spectrum slots, the 2-layer connection link contains 64 spectrum slots, the 3-layer connection link contains 32 spectrum slots, and all connection links are bidirectional. Every server initially has 80 units of computing resources. The values of the parameters used to calculate the power consumption are summarized in Table 2, according to the previous works in [10, 11]. All VN embedding requests are generated with a Poisson traffic model. The number of VMs owned by each VN embedding request is randomly determined within the range from 2 to 4, while the computing requirement of a VM has 4–8 units of computing resources. The connection probability is 50%[2)] for a pair of VMs, and the spectrum requirement of a VLP has 4–8 spectrum slots.

The benchmarks are load-balancing-oriented VN embedding algorithm (LB in short) [2] and VN embedding via sleeping servers (SS in short) [8]. We consider 4 performance metrics: total power consumption (including servers and switches), total switch power consumption, average power consumption per VN embedding request, and average blocking rate (ABR).

4.2 Results and analysis

In Figure 5(a), we generate VN embedding requests from 10 to 100, in order to compare the total power consumption of servers and switches. With the increasing number of VN embedding requests, the total power consumption rises. In the beginning, all curves increase sharply with the consumption of a large

2) There are probably $i(i-1)/4$ VLPs on average for a VN embedding request with i VMs.

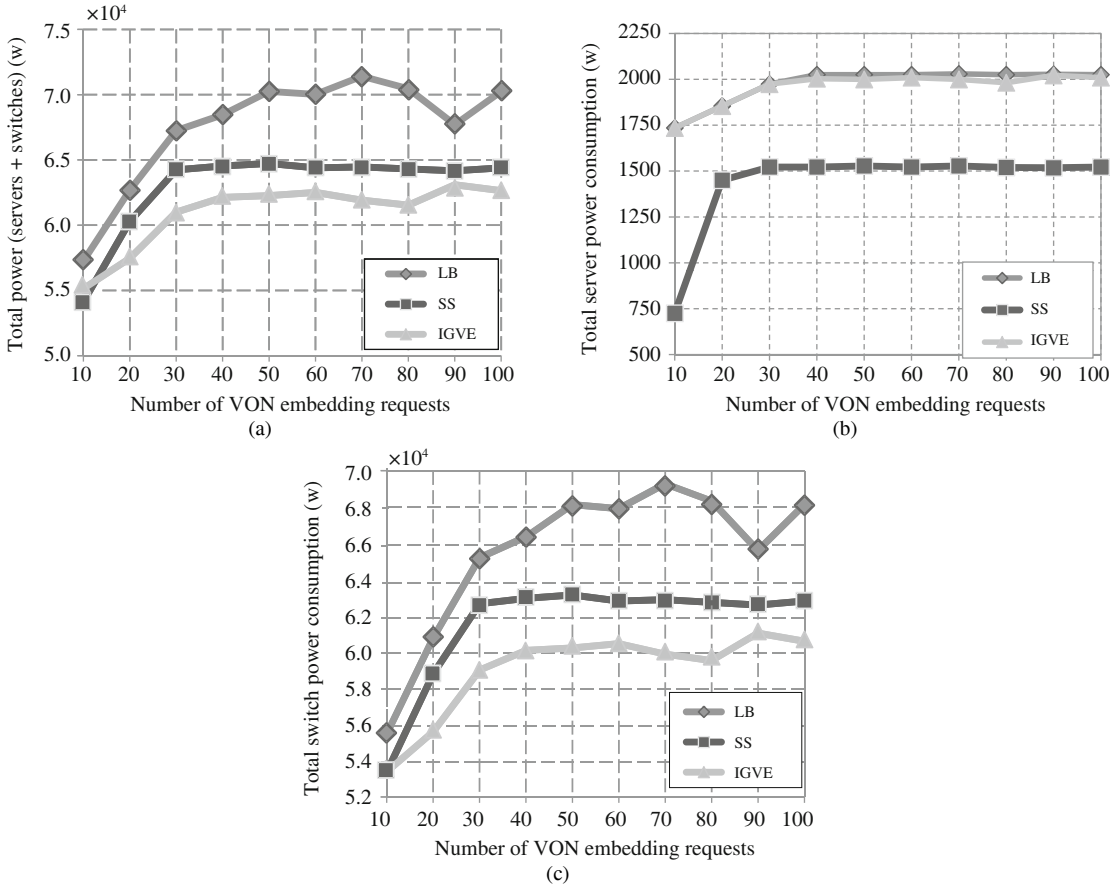

Figure 5 (a) Comparison of the total power consumption; (b) comparison of the total server power consumption; (c) comparison of the total switch power consumption.

number of servers and switches. But then, the variation rate of the total power consumption tends to become smooth since we are not required to turn on many devices for serving consequent VN embedding requests. More importantly, the total power consumption of our IGVE is always the lowest, while the highest one is the benchmark LB. That is because IGVE chooses a single DC or multiple DCs with the short distance between each other, as the target DC(s), and the servers in the same cluster are preferred to hold VMs, which reduces the total power consumption. However, the LB merely considers load balancing, which sometimes is inconsistent with the improvement of power efficiency. Furthermore, compared with LB and SS, the improvement ratios of IGVE for power efficiency are 13% and 5%, respectively.

Figure 5(b) compares the total server power consumption among three algorithms. We can see that with the increasing number of VN embedding requests, three curves rise. The SS has the lowest server power consumption, while LB gets the highest one. That is because SS is the absolute server-sleeping strategy. At the VM mapping stage, SS tends to map VMs into high-load servers. On the contrary, LB tries to map VMs into light-duty servers. Thus, during the same period, LB turns on more servers compared with SS. Our IGVE performs slightly better than LB since it takes server clustering into account. In addition, three curves increase sharply in the beginning, and then they all tend to become smooth.

Though our IGVE does not have the lowest server power consumption, the lowest switch power consumption still guarantees it has the minimal total power consumption. In Figure 5(c), we compare the total switch power consumption among three algorithms. Before discussing, we must point out that the variation tendency of all curves in Figure 5(c) is very similar to that in Figure 5(a). This is a very interesting phenomenon, and the reason of this is the idle power of switches is very high, and many ports are within all types of switches; in contrast, the idle and working power of servers are both small, which

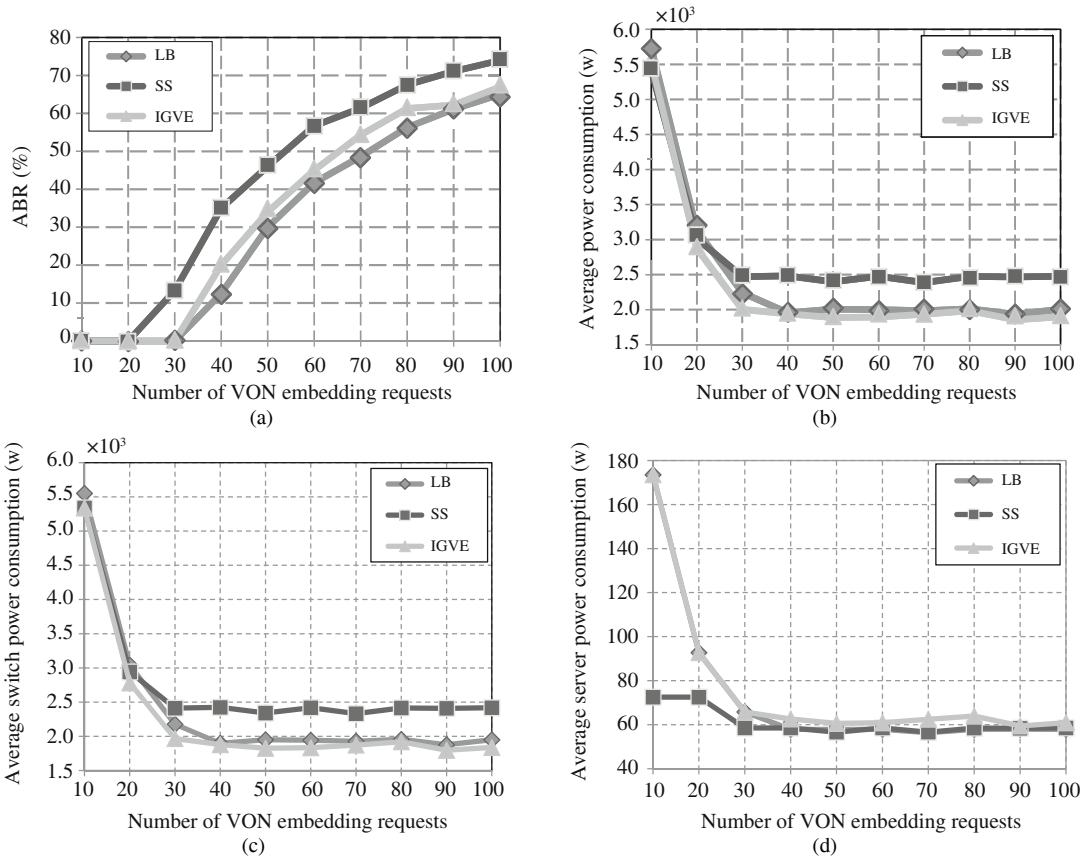

Figure 6 (a) Comparison of ABR; (b) comparison of the average power consumption per VN embedding request; (c) comparison of the average switch power consumption per VN embedding request; (d) comparison of the average server power consumption per VN embedding request.

means that switches play the dominating role in power consumption. IGVE performs the best in terms of saving the power consumptions of switches, because it maintains intra-cluster server lists so that the VM mapping in the same cluster or adjacent clusters reduces the length of substrate paths for VLPs, without consuming many switches.

In Figure 6(a), we compare the ABR among three algorithms. We can see that, with the increasing number of VN embedding requests, the ABR of algorithms rises. The LB performs the best, our IGVE follows, and SS performs the worst. The reason for this is the absolute load balancing in LB is helpful to improve the throughput. Above all, IGVE not only can obtain the higher power efficiency, but also has an acceptable ABR in a load-balancing manner.

When calculating the total power consumption, some VN embedding requests may be blocked under a resource-strained condition. For fairness, we compare the average power consumption per VN embedding request in Figure 6(b). The number of VN embedding requests has a linearly increasing trend and the variation rate is large, but the total power consumption increases in a non-linear way with a small variation rate. Therefore, all curves decrease in Figure 6(b), and the decreasing rate sharply falls down with the increasing number of VN embedding requests initially. Also, IGVE performs the best. The superiority of IGVE over LB in terms of power savings has been discussed. IGVE performs better than SS since servers do not play the dominating role in power consumption though SS saves the power consumed by servers. Figure 6(c) compares the average switch power consumption among three algorithms. The variation trend of three curves is very consistent with that in Figure 6(b). That is because that the power of switches takes majority proportion of the total power consumption.

Figure 6(d) compares the average server power consumption per VN request among three algorithms. With the increasing number of VN requests, all curves decrease. In the beginning, the decreasing trend is

obvious, because the increasing rate of the total server power consumption, both LB and IGVE, cannot catch the increasing rate of VN embedding requests. However, the total server power consumption of SS increases faster than the number of VN requests, consequently, the curve of SS drops gently. With the increasing number of VN requests, three curves become coincident. Therefore, our IGVE has not only the lower switch power consumption, but also the low average server power consumption. In the near future, we will take the preliminary work [12] into our design.

5 Conclusion

The application instantiations of EDCNs, e.g., large-scale cloud computing, 3D cloud rendering, online shopping and payment platforms, grow vigorously, which makes the power consumption draw many concerns in the VN embedding field. The existing power-efficient solutions cannot perform the power savings of switches responsible for the intra- and inter-DC traffic routing. In this paper, we formulated the problem with the integrated objective of saving the power consumed by servers and switches, and designed the heuristic IGVE that consolidates VMs into a single DC or multiple DCs with the short distance between each other. And the servers in the same or adjacent cluster are preferred to hold VMs in IGVE. The simulation results demonstrated that our method was actually effective in terms of power savings and load balancing, and our method had the improvement ratio of about 5%-13% over benchmarks.

Acknowledgements This work was supported in part by Open Foundation of State Key Laboratory of Information Photonics and Optical Communications (Grant No. IPOC2014B009), Fundamental Research Funds for the Central Universities (Grant Nos. N130817002, N140405005, N150401002), Foundation of the Education Department of Liaoning Province (Grant No. L2014089), National Natural Science Foundation of China (Grant Nos. 61302070, 61401082, 61471109, 61502075), Liaoning BaiQianWan Talents Program, and National High-Level Personnel Special Support Program for Youth Top-Notch Talent.

Conflict of interest The authors declare that they have no conflict of interest.

References

1 Yu C Q, Hou W G, Wu Y, et al. Adaptive multilevel modulation for grooming in elastic optical networks. Photon Netw Commun, 2015, doi: 10.1007/s11107-015-0543-x
2 Gong L, Zhu Z Q. Virtual optical network embedding (VONE) over elastic optical networks. IEEE/OSA J Lightwave Technol, 2014, 32: 450–460
3 Cisco. Cisco global cloud index: forecast and methodology. Cisco White Paper, 2011. 2011–2016
4 Kachris C, Kanonakis K, Tomkos I. Optical interconnection networks in data centers: recent trends and future challenges. IEEE Commun Mag, 2013, 51: 39–45
5 Benson T, Akella A, Maltz D. Network traffic characteristics of data centers in the wild. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, New Delhi, 2010. 267–280
6 Taubenblatt M. Optical interconnects for high-performance computing. IEEE/OSA J Lightwave Technol, 2012, 30: 448–457
7 Pepeljugoski P, Kash J, Doany F, et al. Low power and high density optical interconnects for future supercomputers. In: Proceedings of 2010 Conference on Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference (OFC/NFOEC), San Diego, 2010. 1–3
8 Sun G, Anand V, Liao D, et al. Power-efficient provisioning for online virtual network requests in cloud-based data centers. IEEE Syst J, 2015, 9: 427–441
9 Greenberg A, Hamilton J, Maltz D, et al. The cost of a cloud: research problems in data center networks. In: Proceedings of SIGCOMM, Barcelona, 2009. 68–73
10 Aleksic S. Analysis of power consumption in future high-capacity network nodes. IEEE/OSA J Opt Commun Netw, 2009, 1: 245–258
11 Fiorani M, Casoni M, Aleksic S. Large data center interconnects employing hybrid optical switching. In: Proceedings of 18th European Conference on Network and Optical Communications & 8th Conference on Optical Cabling and Infrastructure, Graz, 2013. 61–68
12 Guo L, Cao J N, Yu H F, et al. A new shared-risk link groups (SRLG)-disjoint path provisioning with shared protection in WDM optical networks. J Netw Comput Appl, 2007, 30: 650–661