doi:10.3788/gzxb20144307.0706007

虚拟数据中心的功率有效嵌入算法

闫芳芳,李东,胡卫生

(上海交通大学 区域光纤通信网与新型光通信系统国家重点实验室,上海 200240)

摘 要:针对虚拟化技术中,数据中心能源消耗问题,本文研究功率有效的虚拟数据中心嵌入算法.该算法由功率感知嵌入算法、准入策略算法和碎片整理算法三部分组成.在树形网络拓扑结构中,提出功率感知的嵌入算法,得到虚拟数据中心的最低功耗嵌入方式;在动态虚拟网络业务环境中,一般嵌入算法会造成功率波动,通过业务接入策略减少功率波动;随着业务动态地进入和离开,物理网络中出现计算资源碎片,提出碎片整理算法,通过虚拟机迁移的方法,提高服务器的利用率.仿真和评估证明:随着时间的推移,算法能减少功率开支并有效降低功率波动.

关键词:数据中心; 虚拟化; 嵌入算法; 带宽保障;接入控制

中图分类号:TN915 **文献标识码**:A **文章编号**:1004-4213(2014)07-0706007-6

Power Efficient Allocation of Virtual Data Centers with Bandwidth Guarantee

YAN Fang-fang, LEE Tony-Tong, HU Wei-sheng

(*State Key Laboratory of Advanced Optical Communication Systems and Network Shanghai Jiao Tong University, Shanghai* 200240, *China*)

Abstract: In order to solve the energy consumption in the data centers with virtualization, the power efficient allocation of Virtual Data Centers was investigated which included three components, namely power-aware embedding algorithm, access policy and defragmentation algorithm. A power-aware embedding algorithm was proposed to allocate resources for Virtual Data Centers in tree-like network topologies, and enforce an access policy on the incoming Virtual Data Centers requests to reduce fluctuations in the power expenditure. As requested Virtual Data Centers enter and leave dynamically, there will be resource fragment in the physical network. Therefore, a defragmentation algorithm was devised to migrate virtual machines away from underutilized physical machines. These algorithms were evaluated by simulation and proven effective to smooth and reduce power expenditure over time.

Key words: Data center; Virtualization; Embedding algorithm; Access control

OCIS Codes: 060.4250; 060.4256; 060.4259

0 Introduction

Virtualization has become more and more important over the last years, which enables multiple different applications (e.g. servers, operating systems and sub networks) to run upon the same shared physical resources. Network virtualization has the potential to provide predictable performance for tenants in cloud computing. A novel data center network abstraction was proposed and implemented in a system called Oktopus[1]. The abstraction simplifies the nodes to uniform virtual machines and expresses the bandwidth between virtual machines according to hose model[2]. This abstraction has been adopted for its simplicity and flexibility[2-5]. A greedy algorithm was proposed to allocate virtual data centers in the substrate tree topology[1]. Under a timing varying abstraction of virtual data center, dynamic programming

Foundation item: The National Natural Science Foundation of China(No. 61201223, 61172065) and Shanghai STCSM Science Foundation (No. 12ZR1445600)

First author: YAN Fang-fang (1984—), female, lecturer, PhD degree, mainly focuses on optical networks, packet switching and data center networks. Email: yff@sjtu.edu.cn

Received: Oct. 22, 2013; **Accepted**: Dec. 19, 2013

is employed to search for a feasible allocation[5]. One of the major concerns in data center is energy consumption. Energy aware optimization, virtual machine migration and consolidation[6-9] are green techniques to embed Virtual Data Centers (VDCs) in substrate networks. A method based on mixed integer programming was used to compute an optimal energy efficient embedding solution[6]. If the utilization of a host falls below a threshold, virtual machines will be migrated away to eliminate idle power consumptions[7]. It is shown that a VDC planner can minimize the energy cost for dynamic traffic by using virtual machine migration and consolidation[9].

As consequence of dynamical VDC provision, some servers and links may have fragmented resource and not available for future requests with large bandwidth demand, which is a waste of energy in large data centers. This fragmentation problem resembles the classic dynamic memory allocation problem in operating systems, and the bandwidth fragmentation problem in flexible optical networks[10-11].

Under a greedy allocation algorithm, the network will accept the VDC request whenever an embedding solution is feasible. Thus, any greedy algorithm will lead to low utilization due to peak resource allocation, in particular, when the instant arrival rate is much higher than the mean rate, or the requested resource is far beyond the average demand. In general, the resource utilization will drop sharply upon the departure of large VDC requests. The fluctuation of resource utilization reflects the fluctuation of energy consumption, which can be a major obstacle for energy planning. Even if we can dynamically tune the servers to sleep mode to reduce idle energy expenditure, still some servers have to transit from sleep mode to active mode again in a very short time to serve new arrival requests.

In light of the above concerns, we will investigate energy aware embedding of dynamic VDC requests under the Oktupos abstraction with three components, namely power-aware embedding algorithm, access policy and defragmentation algorithm. Firstly, we propose a power-aware embedding algorithm, which compute an optimal resource allocation solution with lowest power expenditure for VDCs in tree-like network topologies. The power-aware algorithm has the same time and space complexity compared with the baseline dynamic programming algorithm. As consequence of dynamic traffic, there will be resource fragment in the physical network. Therefore, we devise a defragmentation algorithm to migrate virtual machines away from underutilized physical machines. The defragmentation algorithm not only improves the server utilization, but also saves the communication bandwidth by merging Virtual Machines (VMs) in separate servers in a single server. Access control was originally proposed in ATM network and extended to optical network applications[13]. In this paper, we propose an access policy on the incoming VDC requests to smoothing energy usage.

1 Embedding algorithm with defragmentation and access control

A VDC request is defined as a cluster (N, B), where N denotes number of virtual machines and B denotes communication bandwidth, according to the Oktopus abstraction[1]. The abstraction implies that a virtual switch connects these VMs with bandwidth B. The embedding algorithm attempts to place all the VMs in the servers in the substrate data center network. In the same time, the substrate network must have sufficient bandwidth on the paths to support predictable communication between these VMs. In the tree network, the uplink of a servers or switch v is a cutting edge that divides the embedding VMs in two partitions with m and $N-m$ VMs respectively. Then the residual bandwidth of this link $b(v)$ must satisfy the communication requirement for any feasible embedding, that is $b(v) \geqslant \min(m, N-m) * B$[1].

We schedule the process of VDC requests synchronously. In a time slot T_s, the embedding algorithm takes the new arrived requests during the slot and acquires some declined requests in the queue as input. To avoid head-of-line blocking, we implement a look-up queuing discipline, which checks n_s buffered requests for embedding each time slot. A power aware embedding algorithm searches an optimal embedding solution in terms of power consumption for these VDC requests. If none feasible solution exists, a VDC request will wait in the queue and attempt embedding in the next time slots.

We enforce an access control on the requests comparing current power expenditure with the expectation. If the resulting network by the embedding solution is severely over-utilized than the expected power, a request will be rejected temporally and pushed in a queue waiting for better resource condition. Otherwise, the scheduler will accept the VDC request and allocate node and link resources for it. A VDC request can be accepted bypassing the access policy, if the queue length is large than a threshold or its waiting time is larger than the maximal delay. The queue length threshold is introduced to guarantee that the algorithm can track the changes of mean traffic rate quickly.

When the VDC request leaves, the scheduler will release resources assigned to it. The virtual machines of the VDC will be removed from the physical machines.

Consequently, some physical machines are left with one or two VMs, which is called resource fragment. The resource fragmentation in the machines wastes power, because these underutilized machines are running with low loads. In this case, it is beneficial to migrate these VMs to other high utilized physical machines. This function is implemented by the defragmentation component.

1.1 Power-aware embedding algorithm

The embedding algorithm tries to minimize the power consumptions in admitting a VDC request. The energy consumption occurs in servers and switches. As the energy price of provisioning a VDC, some physical machines or switches may have to be reactivated to accommodate new VMs.

Suppose each server can embed E virtual machines at most. The power cost of allocate e VMs of the VDC request on the server u can be computed in Eq. (1), where p_{sleep}, p_{idle} and $p_{\max}$ denote the power in sleep mode, idle mode and full speed mode respectively[7-8]. Here each embedded VM is assumed to work at full CPU speed. Thus we take e/E as the CPU utilization in Eq. (1). An idle server will transit automatically with very short latency (less than 1 second[14] and is neglected in the simulation) to sleep mode and consume very low power of p_{sleep}.

$$p(e,u)=\begin{cases} p_{\text{idle}}+\dfrac{(p_{\max}-p_{\text{idle}})e}{E}-p_{\text{sleep}}, u \text{ was in sleep} \\ \dfrac{(p_{\max}-p_{\text{idle}})e}{E}, \text{otherwise} \end{cases} \quad (1)$$

We assume all ports in the switches are enabled when turned-on. The power in an active switch is approximately constant regardless of traffic load, as going from zero to full traffic increases power by less than 8% [15]. The power cost of connecting virtual machines via a switch v in level l is expressed in Eq. (2), where p^l_{sleep} and $p^l_{\max}$ denote the power costs in the sleep mode and active mode respectively.

$$p(v)=\begin{cases} p^l_{\max}-p^l_{\text{sleep}}, v \text{ was in sleep} \\ 0, \text{otherwise} \end{cases} \quad (2)$$

Dynamic programming is used in Ref. [5] to find a feasible embedding, and in Ref. [4] to minimize congestion in a binary tree. Hence we adopt dynamic programming to search for the optimal embedding solution in tree-like topology, shown in Fig. 1. The variables used in the power aware algorithm are defined here:

$m_v(k)$: the set containing all feasible numbers of VMs that can be embedded in the first k children of subtree v, which is initiated as Φ.

$d_v(h,k)$: the number of VMs that should be assigned to the k^{th} children given the configuration that embed h VMs totally in the first k children of subtree v, which is initiated as 0.

$p_v(h,k)$: the power cost of embedding h VMs in the first k children of subtree v, which is initiated as ∞.

$M(v)$: the set containing all feasible numbers of VMs that can be embedded in the subtree v, which is initiated as Φ.

$P(h,v)$: the power cost of embedding h VMs in the subtree v, which is initiated as ∞.

$b(v)$: residual bandwidth in the uplink of subtree v.

Power aware embedding algorithm

Input: a VDC request (N,B), tree topology of data center

1. **For** level $l=1$ to height of Tree
2. **For** each subtree v in the level l
3. **For** each child $k=1$ to $|v|$ of the subtree v
4. Initialize: $m_v(k) = m_v(k(1)$;
5. $d_v(h,k)=0$; $p_v(h,k)=p_v(h,k-1)$ for each $h\in m_v(k-1)$;
6. Compute $P(e,v_k)$ by (1) if v_k is a server.
7. **For** each $h\in m_v(k-1), e\in M(v_k)$
8. **If** $e+h\notin m_v(k)$ *or* $p_v(e+h,k)>p_v(h,k-1)+P(e,v_k)$
9. $m_v(k) = m_v(k)\cup e+h$
10. $d_v(e+h,k)=e$
11. $p_v(e+h,k)=p_v(h,k-1)+P(e,v_k)$
12. **For** each $h\in m_v(|v|)$, verify the bandwidth constraint:
13. **If** $\min(h,N-h)*B\leqslant b(v)$
14. $M(v)=M(v)\cup h$
15. $P(h,v)=p_v(h,|v|)$
16. **If** the root switch of subtree v is re-activated
17. $P(h,v)= P(h,v)+p^l_{\max}-p^l_{\text{sleep}}$ for all h
18. If the set of feasible subtrees $V_{(N,B)}=\{v|\ N\in M(v)\}\neq\Phi$
19. Find the subtree that minimizes power consumption, i. e.
20. $\hat{v}=\min\{\ v\in V_{(N,B)}|P(h,v)\}$
21. **If** $P(h,\hat{v})+W(t)\leqslant\hat{W}(t)$
22. Allocate the request (N, B) in the subtree according (1) to $d_v(h,k)$ *with* $k=|\hat{v}|$ to 1
23. **Else**
24. Store the request in the queue
25. **Return**

Fig. 1 Power aware embedding algorithm

The algorithm prefers to embed the VMs in the lower index children in a subtree, as declared in the initiation step. The k^{th} (higher index) child is added to the solution, only when it can include more VMs or consume less energy. Among all the feasible solutions in the same level, we select the subtree which contributes to minimum power.

For each child of the subtree v, the dynamic programming takes time of $O(N^2)$ for requests with N virtual machines. Therefore, the power-aware algorithm has time complexity of $O((|V_s|+|V_m|)*N^2)$, which is the same with the baseline dynamic programming algorithm [5], where V_s and V_m denote the number of switches and servers respectively in the substrate data

center network. Although two additional matrices, $p_v(h,k)$ and $P(h, v)$, are needed for store the temporary power cost, the power-aware algorithm has the same space complexity of $O(d * N)$ with the dynamic programming algorithm, where d represents the maximal degree of the substrate tree network.

1.2 Access policy

Let $W(t)$ denote the actual power expenditure at time t. The access controller samples the power expenditure $W(t)$ uniformly at intervals of T_s. It computes the average power expenditure $W_\mu(t-\tau,t)$ and its standard variance $W_\delta(t-\tau,t)$ in the time duration $(t-\tau,t)$. The predicted power expenditure at time t can be computed in Eq. (5), where the constant $\alpha=1$. The optimal embedding result will be accepted, if the resulting power consumption is less than the predicted power expenditure, which is shown in line 21～24 of Fig. 1.

$$W_\mu(t-\tau,t)=\frac{1}{\tau/T_S}\sum_{i=1}^{\tau/T_S}(W(t-i*T_S) \tag{3}$$

$$W_\delta(t-\tau,t)=\sqrt{\frac{1}{\tau/T_S}\sum_{i=1}^{\tau/T_S}(W(t-i*T_S)-W_\mu(t-i*T_S,t))^2} \tag{4}$$

$$\hat{W}(t)=W_\mu(t-\tau,t)+\alpha W_\delta(t-\tau,t) \tag{5}$$

1.3 Defragmentation algorithm

As the result of dynamical VDC provision, some machines may have fragment resource and not available for future requests with large bandwidth demand. The defragmentation component will first mark the existing VDCs that reside on the machines of utilization lower than a threshold. The defragmentation algorithm tries to migrate a number of VMs away from underutilized servers to other servers. Migration between high utilized servers is useless and forbidden in the defragmentation algorithm. These VDCs will be reconfigured, if destination servers for migration are successfully found.

The defragmentation problem can also be solved with dynamic programming. For the VDC request (N, B), assume that m VMs from n_L underutilized machines need migration to n_H high utilized machines in the substrate network. The algorithm will find out how to embed m more VMs in the n_H machines. The defragmentation algorithm is shown in Fig. 2. The variables used in the defragmentation algorithm are defined here:

$m'_v(k)$: the set containing all feasible numbers of VMs that can be migrated to the first k children of subtree v, which is initiated as Φ.

$d'_v(h,k)$: the number of VMs that should migrated to the k^{th} children given the configuration that migrate h VMs totally to the first k children of subtree v, which is initiated as 0.

$M'(v)$: the set containing all feasible numbers of VMs that can be migrated into the subtree v, which is initiated as Φ.

$N'(v)$: number of VMs embedded the high utilized servers in the subtree v according to the current mapping of the VDC.

$B'(v)$: bandwidth shares that are allocated to the VDC in the uplink of subtree v according to the current mapping.

Defragmentation algorithm

Input: a VDC request that requires migration (N,B,m).

1. **For** level $l=1$ to height of tree
2. **For** each subtree v in the level l (Consider the subtrees that currently embed some VMs of the VDC first)
3. **For** each child $k=1$ to $|v|$ of the subtree v
4. Initialize: $m'_v(k)=m'_v(k(1)$; $d'_v(h,k)=0$ for each $h\in m'_v(k-1)$
5. **For** each $h\in m'_v(k-1), e\in M'(v_k)$
6. If $e+h\notin m_v(k)$
7. $m'_v(k)=m'_v(k)\bigcup e+h$
8. $d'_v(e+h,k)=e$
9. **For** each $h\in m'_v(|v|)$, verify the bandwidth constraint:
10. **If** $\min(h+N'(v),N-h-N'(v))*B\leqslant b(v)+B'(v)$
11. $M'(v)=M'(v)\bigcup h$
12. **If** $m\in M'(v)$ and $m+N'(v)=N$, then we have found a feasible migration solution,
13. **Reallocate** the request (N,B,m) in the subtree v according to $d'_v(h,k)$ with $k=|v|$ to 1
14. **Return**

Fig. 2 Defragmentation algorithm

The algorithm performs migrations from underutilized servers to other servers in the same VDC with high priority. Consequently, VMs in separate servers can be merged in a single server, which will save the communication bandwidth between these VMs. The VMs are migrated to other servers which do not host any VMs of the VDC, only if necessary. Let V' be the smallest subtree that already host all N VMs according to the current mapping. Then the algorithm can be terminated, if all the m VMs have found their destination machines for migration after T' is traversed. Otherwise, the algorithm has to search for destination servers for migration outside the original subtree T'. We do not calculate power cost in the defragmentation algorithm, because none machines are re-activated.

For each child of the subtree v, the dynamic programming takes time of $O(m^2)$ for requests with m virtual machines to be migrated. The defragmentation algorithm has time complexity of $O((|V_s|+|V_m|)*m^2)$, which is lower than the power-aware embedding algorithm. Similarly, the defragmentation algorithm has space complexity of $O(d*m)$, which is lower than both the dynamic programming algorithm[5] and the power-

aware embedding algorithm.

Apparently, migration is associated with cost. We define $\beta(x_i, y_i)$ as the cost of migration a VM x_i to a potential destination server y_i, which calculate the hops of the migration path. Let π be the final migration plan for the m VMs. Then the migration cost problem is to minimize $\sum_{i=1}^{m}\beta(x_i, \pi(x_i))$.

We can build a bipartite graph for the migration cost problem. View each VM to be migrated as an ingress vertex, while a destination server as an egress vertex. An edge with weight $\beta(x_i, y_i)$ connects the vertices corresponding to VM x_i and destination y_i. Therefore, the minimum cost is achieved by minimum weight matching in the bipartite graph.

2 Performance evaluation

We conducted simulation on a two-tier data center network of tree topology. Each server has equal capacity that supports 4 VMs. A server at sleep, idle and full speed has power of 10 W, 175 W(70% of max) and 250 W. A top-of-rack (ToR) switch connects 10 servers by 1 Gbps link. A core switch links 20 ToR switches by 10 Gbps link. To R and core switches have power of 100 W and 200 W respectively in the active mode, and both have power of 10 W in the sleep mode.

We generate dynamic VDC requests according to Poisson arrival process. The holding time of each VDC obeys exponential distribution with mean of 1 hour. The number of VMs of each VDC N is uniform distributed from 10 to 30. The bandwidth requirement B has exponential distribution with mean of 1 Mbps. The queue length threshold is set to be 10, while the maximal delay is set to be 3 hours.

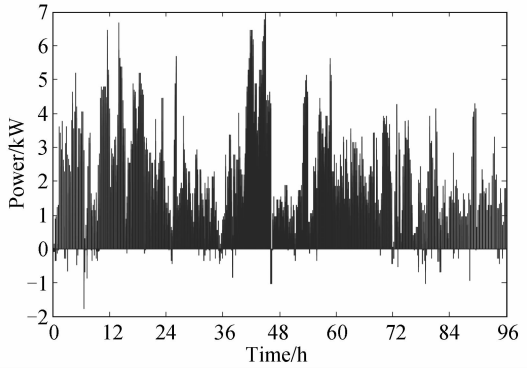

Fig. 3 Power reduction of power aware embedding algorithm with defragmentation over baseline algorithm

In the first experiment, the VDC request arrives at a constant rate 25/h all day. The dynamic programming algorithm in Ref. [5] is referred as baseline algorithm. We evaluate the power reduction of power-aware embedding algorithm with defragmentation compared to baseline algorithm, which is shown in Fig. 3. The power reduction is mostly positive, except for very few cases. The maximal power reduction is 7 kW, which accounts for the power expenditure of 28 servers. The mediate power reduction is 3.5 kW, approximately 10% of the mediate power expenditure of the whole data center network shown as in Fig. 4.

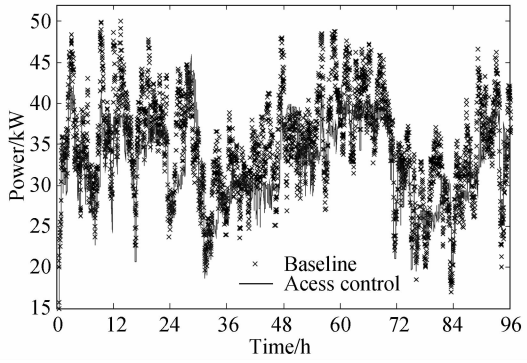

Fig. 4 Smooth the power fluctuation for constant arrivals with access control

The time duration to compute average and standard variance in (2)-(4) for access control is set to be 2 h. We evaluate the impact of access control on the live power expenditure, which is depicted in Fig. 4. The peak power of baseline algorithm is more than 50 kW, which means almost all the machines and switches run at full power. The peak power of access control is reduced to 45 kW, while most of the time the whole network consumes less than 40 kW. The power fluctuation is significantly smoothed. For example, the power expenditure remains stable below 40 kW in the day-3.

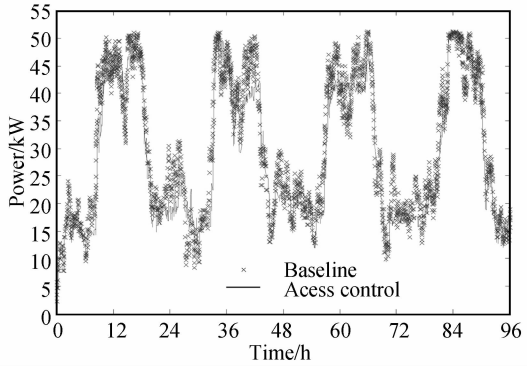

Fig. 5 Smooth the power fluctuation for time-varying arrivals with access control

We also test the effectiveness of access control under the circumstance of piecewise traffic rate, shown in Fig. 5. In the second experiment, the VDC request arrives at a higher rate 30/h in working day time from 8 : 00 to 18 : 00; and at a lower rate 15/h in the night, while resembles the real traffic. We find that the algorithm can track the traffic steps without noticeable delay. The access control can smooth the live power

expenditure under both heavy and light load.

The new arrival requests that violate power threshold will endure additional delay before access the data center again. The delay of baseline algorithm and access control is shown in Fig. 6. Less than 5% requests are delayed, because the servers and bandwidth are fully occupied at their arrivals. About 30% requests experience additional delay as a penalty of smoothing power expenditure, rather than resource deficiency.

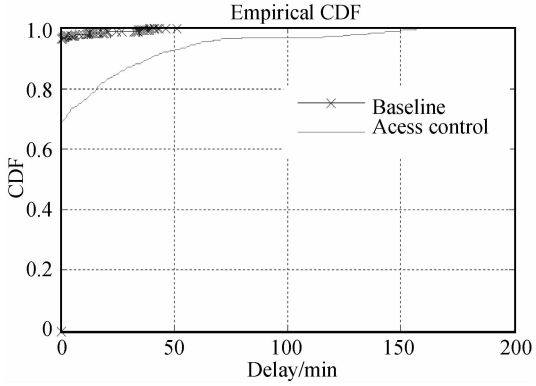

Fig. 6 Delay penalty of access control

3 Conclusion

This paper proposed a scheduling system for power efficient allocation of virtual data centers (VDCs) including three components, power-aware embedding algorithm, access policy and defragmentation algorithm. A power-aware embedding algorithm was proposed to compute the optimal embedding for VDCs with lowest power cost in tree-like network topologies. An access policy was employed to reduce fluctuations in the power expenditure. Moreover, a defragmentation algorithm was devised to migrate virtual machines away from underutilized physical machines. The time and space complexity of power-aware and defragmentation algorithms were analyzed. The algorithms were evaluated by extensive simulation and proven effective to stabilize and reduce power expenditure over time.

The power-aware and defragmentation algorithms proposed in this paper assume a tree topology of the substrate network. In the tree topology, any link cuts the network into two separate partitions and there is only a single path between any two servers, which makes the embedding problem solvable within polynomial time. The access policy is a general rule for smoothing the power expenditure and thus not restricted to specific substrate network topology and virtual network abstraction. Therefore, how to extend the embedding and defragmentation algorithm to general topology with multipath routing is worth further exploration.

Reference

[1] BALLANI H, COSTA P, KARAGIANNIS T, *et al*. Towards predictable datacenter networks[C]. SIGCOMM, 2011, 242-253.

[2] ZHU Jing, LI Dan, WU Jian-ping, *et al*. Towards bandwidth guarantee in multi-tenancy cloud computing networks[C]. IEEE ICNP 2012, 1-10.

[3] BALLANI H, GUNAWARDENA D, KARAGIANNIS T. Network sharing in multi-tenant datacenters[R]. Technical Report of Microsoft Research, MSR-TR-2012-39, 2012, 1-15.

[4] DUTTA D, KAPRALOV M, POST I. Optimal bandwidth-aware VM allocation for Infrastructure-as-a-Service[R]. CoRR abs/1202.3683, 2012, 1-12.

[5] XIE Di, DING Ning, HU Y, *et al*. The only constant is change: incorporating time-varying network reservations in data centers [C]. SIGCOMM, 2012, 199-210.

[6] BOTERO J, HESSELBACH X, DUELLI M, *et al*. Energy efficient virtual network embedding[J]. *IEEE Communications Letters*, 2012, **16**(5): 756-758.

[7] BELOGLAZOV A, ABAWAJY J, BUYYA R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing[J]. *Future Generation Computer Systems*, 2012, **28**(5): 755 - 768.

[8] SU Sen, ZHANG Zhong-bao, CHENG Xiang, *et al*. Energy-aware virtual network embedding through consolidation[C]. INFOCOM, 2012, 2708-2713.

[9] ZHANI M, ZHANG Qi, SIMON G, *et al*. VDC planner-dynamic migration-aware virtual data center embedding for clouds[C]. Proc. of the IFIP/IEEE Integrated Network Management Symposium (IM), 2013, 1-8.

[10] JU Wei-guo, HUANG Shan-guo, XU Zhen-zhen, *et al*. Spectrum fusion oriented routing and spectrum allocation algorithm and spectrum defragmentation algorithm [J]. *Acta Photonica Sinica*, 2013, **42**(8): 929-935.

[11] CHEN Cun-kang, QIAO Yao-jun, Ji Yue-feng, Dynamic bandwidth allocation algorithm for orthogonal frequency division multiplexing access passive optical network[J]. *Acta Photonica Sinica*, 2011, **40**(5): 684-689.

[12] DUELD N, GOYAL P, GREENBERG A, *et al*. A flexible model for resource management in virtual private networks [C]. SIGCOMM, 1999, 95-108.

[13] QIAO Yao-jun, DING Fu-ling, JI Yue-feng, Media access control protocol design in OFDM-PON[J]. *Acta Photonica Sinica*, 2013, **21**(6): 654-660.

[14] MEISNER D, GOLD B, WENISCH T. PowerNap- eliminating server idle power[C]. Proceedings of the 14th international conference on Architectural support for programming languages and operating systems, 2009, 205-216.

[15] HELLER B, SEETHARAMAN S, MAHADEVAN P. ElasticTree: saving energy in data center networks[C]. Proceedings of the 7th USENIX conference on Networked systems design and implementation (NSDI), 2010, 1-16.