Journal of Interconnection Networks
Vol. 14, No. 3 (2013) 1360002 (17 pages)

DOI: 10.1142/S0219265913600025

Differentiated Bandwidth Guarantees for Cloud Data Centers

YANGYANG LI*, HONGBO WANG, JIANKANG DONG,
JUNBO LI and SHIDUAN CHENG

State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications,
Beijing 10086, P. R. China
yyli@bupt.edu.cn*
hbwang@bupt.edu.cn
dongjk@bupt.edu.cn
lijunbo@bupt.edu.cn
chsd@bupt.edu.cn

Received 30 April 2013
Revised 29 November 2013

By means of virtualization, computing and storage resources are effectively multiplexed by different applications in cloud data centers. However, there lacks useful approaches to share the internal network resource of cloud data centers. Invalid network sharing not only degrade the performance of applications, but also affect the efficiency of data center operation. To guarantee network performance of applications and provide fine-grained service differentiation, in this paper, we propose a differentiated bandwidth guarantee scheme for data center networks. Utility functions are constructed according to the throughput and delay sensitive characteristics of different applications. Aiming to maximize the utility of all applications, the problem is formulated as a multi-objective optimization problem. We solve this problem using a heuristic algorithm: the elitist Non-Dominated Sorted Genetic Algorithm-II(NSGA-II), and we make a multi-attribute decision to refine the solutions. Extensive simulations are conducted to show that our scheme provides minimum bandwidth guarantees and achieves more fine-grained service differentiation than existing approaches. The simulation also verifies that the proposed mechanism is suitable for arbitrary data center architectures.

Keywords: Cloud computing; data center network; application utility; bandwidth guarantee.

1. Introduction

In recent years, cloud computing has been considered as one of the revolutionary technology by both industry and academia. According to the definition from UC

*79 Maibox, No. 10 Xitucheng Road, Haidian District, Beijing 100876, China.

Berkeley,[5] cloud computing refers to *the applications delivered as pay-as-you-go services over the Internet, as well as the hardware and system software in the data centers that provide those services*. Beyond a doubt, the crucial component in data centers is the internal network which inter-connects hardware and services, named data center network (DCN).

To achieve the economics of scale, nowadays, cloud data center mostly offers on-demand computing and storage resources by utilizing virtualization technology[8,18] to multiplex these resources. However, the use and management of network bandwidth in cloud data centers fall short. Recent measurement and analysis[9,17] indicate that, bandwidth is becoming a significant factor which plays an important role in the performance of data centers. The authors of Ref. 6 conducted several experiments in both public data centers and production data centers. They found that network performance varies significantly across applications. Current mechanism cannot guarantee predictable performance of applications which tightly depends on the underlying network. Furthermore, network sharing without guarantees opens door for malicious applications, since they can dominate network resource via deliberately designed TCP variants. The fundamental reason is that TCP can only provide flow-based fairness, however malicious applications could preempt the bandwidth resource by generating more TCP connections than legal ones.

Owing to these drawbacks, we argue that the data center network sharing should satisfy the following three key objectives. First of all, *minimum bandwidth guarantees*. To guarantee application works well even under the worst cases, data center operator should promise to meet the minimum bandwidth requirement for each application. It is also a minimum service level arrangement (SLA) that must be satisfied. If an application's request could not be met, it will be rejected by admission control agent. Second, *fine-grained service differentiation*. Tenants rent servers in data centers to run variety applications, this diversity gives a chance for cloud providers to offer differentiated service levels according to requirements of different applications. As a result, application performance can benefit from differentiated bandwidth guarantees and cloud provider can improve the revenue. The last but not the least, *adaptability*. Bandwidth guarantee mechanism should be suitable for arbitrary architectures of data center networks, the deployment of the scheme should be independent of specific network topology.

Many network sharing schemes[6,13,21,22,25,27] have been proposed recently, nevertheless, none of them can achieve all above goals. Some of them[6,13] cannot improve network bandwidth efficiency as the bandwidth in their schemes are static maximum guaranteed. Expensive bandwidth resource would be wasted if the bandwidths are oversubscribed. The others[21,22,25,27] allocate bandwidth to each tenants, applications or virtual machines based on weights. However, those weight-based methods do not have minimum bandwidth guarantees, network performance of applications cannot be predictable. As a result, the SLA would be violated. Furthermore, the granularity of weight-based guarantees they proposed is too coarse-grained. Applications

are running in the data center vary from data-intensive to user-facing Web services. The former class is more sensitive to throughput whereas the latter is more sensitive to delay. Therefore, all these proposed approaches do not consider these characteristics of applications.

Since application utility is influenced by both throughput and delay. In this paper, we propose an application utility-based model for sharing data center networks. The primary contribution of this paper is *the design of fine-grained differentiated bandwidth guarantee mechanism which combines the features of cloud data centers*. First, to further improve the network efficiency, we allocate bandwidth on multiple paths for each application. Second, to provide fine-grained service differentiation, we construct utility functions based on the throughput and/or delay sensitive characteristics of different applications. Finally, we formulate the bandwidth guarantee problem as a multiple-objective optimization problem, we solve the problem by using a genetic algorithm and obtain the optimal solution based on making a multiple attribute decision. Numerical simulations indicate that our scheme satisfies the predetermined three goals.

The remainder of this paper is organized as follows. After presenting related work in Section 2, we describe the architecture and formulate the problem in Section 3 and Section 4 respectively. Then we give the solution in Section 5 and conduct extensive simulations in Section 6. Finally, we conclude the paper in Section 7.

2. Related Work

In this section, we divide work most relevant to ours into two categories: static bandwidth guarantees and weight-based guarantees. Then we introduce them respectively.

2.1. *Static bandwidth guarantees*

Static bandwidth guarantee means bandwidths are reserved by administrators. There are two examples. One is SecondNet[13] which offers three priority bandwidth guarantees, including type 0, type 1 and best effort. Type 0 provides fixed reserved bandwidth between virtual machines, which is analogous to Integrated Service.[11] Type 1 provides only last and/or first hop guarantee, and best effort type does not have any guarantee. Another one is Oktopus[6] proposed two class virtual network abstractions to meet different application requirements. They distinguished data-intensive applications from others, resulting in Virtual Cluster and Virtual Oversubscribed Cluster are abstracted respectively. Both of the two class abstraction can guarantee fixed switch-to-VM bandwidth. The only difference is that the latter is interconnected with an oversubscription factor. However, even for the highest priority application, in other words, priority 1 in Ref. 13 or "Virtual Cluster" in Ref. 6, the bandwidth each virtual machine obtained is bounded by a fixed value. Applications cannot benefit from extra spare network resource in data centers. What's more, if the

fixed bandwidth is over-subscribed, tenants would lose money which they invested for applications and spare network resources are wasted at the meanwhile.

2.2. *Weight-based bandwidth guarantees*

NetShare[21] and Seawall[27] allocate bandwidth according to the weights of applications in centralized and distributed way respectively. The pure weight-based approaches cannot guarantee predictable performance under the worst cases. Suppose a scene that a great number of applications compete for one congestion link. In this situation, even the application with the largest weight cannot acquire a minimum bandwidth guarantee.

Terry *et al.* bridged this gap in an extended version[22] of NetShare[21] via introducing an admission control strategy. Nevertheless, we argue that the weight-based methods cannot give fine-grained differentiation to different types of applications. Because those approaches only differentiate the bandwidth requirements of applications. However, as before mentioned, variety of applications are running in data centers, they vary from throughput-sensitive to delay-sensitive. Delay sensitive applications prefer to use the least congested path rather than having a larger bandwidth guarantees. In Ref. 25, the authors proposed a network resource sharing scheme which allocates bandwidth in proportion to the payments of tenants, the nature of the sharing is still based on weights, where the payment is equivalence to weights.

The authors of Gatekeeper[26] summarized partial literature aforementioned. They bounded the maximum bandwidth impact with a bandwidth capping and they implemented a prototype on Xen[8] using Openvswitch.[2] However, as their fundamental assumption is that the links between all servers are un-blocking, this corresponds less with reality in current data centers hence limits their usage. Hitesh *et al.*[7] jointed the bandwidth guarantee with virtual machine placement. More peculiarly, they emphasized the communication for VM across tenants. Vimalkumar *et al.*[15] designed a practical edge-based network performance isolation mechanism for multi-tenant cloud, we think it is a perfect complement to the implementation of our scheme.

As far as we can see, this paper is the first one consider both bandwidth and delay needs of different applications. Delay sensitive differentiations give more fine-grained bandwidth guarantees. We leverage the multi-path feature of DCN and make full use of idle bandwidth as much as possible. Compared to bandwidth reserve methods in,[6,13] our scheme has higher bandwidth utilization. We construct utility functions to reflect both throughput-sensitive and delay-sensitive features of applications, hence we believe that taking delay factor into consideration can get more fine-grained service differentiation than those in Refs. 21, 22, 25, 27. In the meanwhile, our scheme can be used without the limitation in Ref. 26.

3. Overview

Centralized controller is commonly used in data center networks for allocating IP addresses and other operations. Taking advantage of this controller, we design the

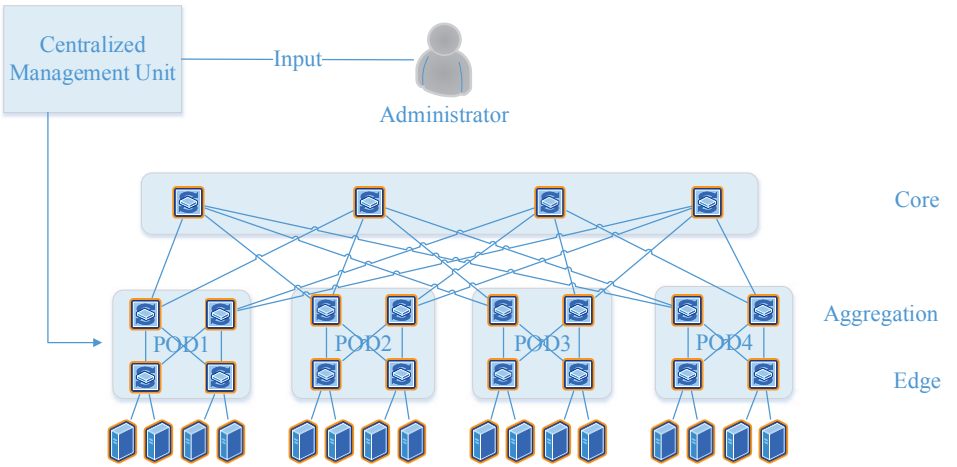


Fig. 1. A simple fat-tree topology.

bandwidth guarantee mechanism in a centralized way and describe the architecture of our scheme in this section.

For convenience, we take one of the well-known DCN topology:[4,12,23] fat-tree[23] as an example to present our scheme. It should be noted that the deployment of our scheme is independent of DCN topologies. As shown in Figure 1, fat-tree is split into three layers, which are labeled edge(access), aggregation and core respectively. There are k Pods, each Pod contains two layers of $k/2$ switches. Each k-port switch in the edge layer is directly connected to $k/2$ servers. The remaining $k/2$ ports are connected to $k/2$ aggregation layer switches. There are $(k/2)^2$ k-port core layer switches. Each core layer switch has one port connects to one of aggregation layer switch in each Pod. In general, a fat-tree builds with k-port switches supports $k^3/4$ servers. In Figure 1, $k = 4$, so it can support up to 16 servers.

As shown in Figure 1, centralized management unit (CMU) is the key component of our mechanism. The main function of CMU is to maintain routing matrix (RM) and compute bandwidth allocation according to the requirements of applications which are input by administrator of data centers. Application requirements should at least contain the following three parameters: throughout-sensitive coefficient, delay-sensitive coefficient and minimum bandwidth requirement. The specific meaning of these parameters we will introduce in the next section.

In order to generate the routing matrix, we need symbols to distinguish each physical links. We label all links in the topology using a similar method which Radhika *et al.*[23] used to allocate IP addresses to switches. In brief, start from Pod 1, we mark the links between edge switches and aggregation switches with number $1 - k/2$ (from left to right). Then the links between aggregation layer and core layer can be labeled in the same way. The remaining links in other Pods could be labeled continually. In Section 6, we will give an example of labeled topologies.

In virtualized cloud data centers, each packet is processed by a virtual switch before the packet be sent out or be forwarded to virtual machines. Virtual switch is usually be implemented in the virtualization layer of physical servers. As a software switch, we can develop some specific functions according to demands. We utilize virtual switches to allocate bandwidth on multiple paths as soon as bandwidth allocation results are received from CMU. The major technologies be applied here are rate-limiting and multi-path routing. Detailed discussion can be found in section 5.

4. Differentiated Bandwidth Guarantees

4.1. *System model*

We model DCN topology as a weighted undirected graph and denote it by $G = (N, L)$, where N is the set of switches and L is the set of physical links denoted by $L = 1, 2, ..., l(l \geq 2)$. The bandwidth capacity and residual capacity of links are denoted by vector $C = (c_1, c_2, ..., c_l)(l \geq 2)$ and $\gamma = (\gamma_1, \gamma_2, ..., \gamma_l)(l \geq 2)$ respectively. A tenant usually rents a group of virtual machines to run their applications. We use VM-VM pair $[srcVM, dstVM]$ to represent the communication between virtual machines. Data center networks are always constructed with multi-root tree topology, virtual machines can communicate with each other using multiple paths. For example, in fat-tree topology, the quantity of the paths for inter-Pod and intra-Pod communication is determined by the number of core switches and aggregation switches in each Pod respectively. We use p_i to represent the number of paths that is available for VM-VM pair i. Then the bandwidth allocated to the i_{th} VM-VM pair can be denoted by vector $X_i = (x_{i1}, x_{i2}, ..., x_{ip_i})(i \geq 2)$, x_{ij} denotes bandwidth be allocated to pair i on path j. We assume that all applications running in the data center use n VM-VM pairs, the global bandwidth allocation vector can be acquired with $X = (X_1, X_2, ..., X_n)(n \geq 2)$. Accordingly, routing matrix can be denoted by

$$R_{l,p} = \begin{pmatrix} R_{1,1} & R_{1,2} & \dots & R_{1,p} \\ R_{2,1} & R_{2,2} & \dots & R_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ R_{l,1} & R_{l,2} & \dots & R_{l,p} \end{pmatrix} \tag{4.1}$$

where

$$P = \sum_{i=1}^{n} p_i \tag{4.2}$$

The number of rows means that there are l physical links in data center network. And p columns mean there are n VM-VM pairs each with p_i redundant paths. The value of elements in the matrix can be determined by using the following indicative

function:

$$R_{i,j} = \begin{cases} 1, if\ link\ i \in path\ j \\ 0, if\ link\ i \notin path\ j \end{cases} \tag{4.3}$$

4.2. *Service model*

For fine-grained differentiated bandwidth guarantees, we propose an application utility-based bandwidth guarantee interface whereby tenants can specify application performance requirements according to the characteristics of applications. The interface can be denoted by using a set of rules of the format $[ApplicationID, srcVM, dstVM, B_{min}, \alpha, \beta]$. Wherein the interface, B_{min} denotes the minimum bandwidth requirement of the application, α and β means the throughput-sensitive and delay-sensitive coefficient of the application respectively. For example, $[app_0, vm_0, vm_1, 10, \alpha, \beta]$ specifies that communication between vm_0 and vm_1 in app_0 requires at least 10 unit bandwidth guarantee with utility coefficient $[\alpha, \beta]$.

As Amazon Elastic Compute Cloud[1] specifies tens of instances for differentiating requirements of CPU, memory, I/O resources. We differentiate network resource requirements by setting the combination of $[\alpha, \beta]$ for different applications. A possible solution is to define $\alpha, \beta \in \{x | 1 \leq x \leq 10, x \in Z\}$, then the combination of $[\alpha, \beta]$ can express bandwidth differentiation for *100* different application types.

Based on the interface we defined above, we can construct utility functions to show the network performance for different types of applications:

$$U_k = \sum_{u:u \in pair(k)} \sum_{v:v \in path(u)} \sum_{w:w \in link(v)} (\alpha_k \chi_{kw} - \frac{\beta_k \chi_{kw}}{\gamma_w}) \tag{4.4}$$

$pair(k)$ denotes the set of VM-VM pairs which belong to application k. $path(u)$ denotes the set of paths used by VM-VM pair u. And $link(v)$ denotes the set of links used by path v. χ_{kw} denotes bandwidth allocate to application k on link w, it can be obtained using bandwidth allocation vector X and routing matrix R. The term $1/\gamma_w$ denotes the expected congestion delay on link w from an $M/M/1$ delay function.[14,19,20,24] α_k and β_k reflect the throughput and delay sensitive characteristic of application k respectively. Hence the utility of application k is consisted of the utility which the application obtained from all links, paths and VM-VM pairs it uses.

The meaning behind the utility is a tradeoff between income brought by bandwidth increases and expenditure charged by congestion delay growth. For a throughput-sensitive applications with no delay requirement can be modeled using $\alpha_k > 0$ and $\beta_k = 0$. Whereas delay-sensitive applications e.g., VOIP applications, can be modeled using $\alpha_k = 0$ and $\beta_k > 0$. Likewise, a variable bandwidth with delay requirement can be modeled using both $\alpha_k > 0$ and $\beta_k > 0$. Specific benchmarks

can be conducted to assist setting the values $[\alpha, \beta]$. However, how to conduct the benchmark tests is out of the scope of this paper.

In Table 1, the key notations which are used throughout the paper are summarized.

Table 1. Key notations in the system model.

Symbol	Description
l	Number of physical links
n	Number of VM-VM pairs
c_i	Bandwidth capacity of physical link i
γ_i	Residual bandwidth of physical link i
X_i	Bandwidth allocated to VM -VM pair i
B^k_{min}	Minimum bandwidth requirement of application k
α_k	Throughput-sensitive coefficient of application k
β_k	Delay-sensitive coefficient of application k
U_k	Utility function of application k

4.3. *Formulation*

The objective of differentiated bandwidth guarantee problem is to find an optimal bandwidth allocation which can maximize the utility of all applications. Hence the problem can be formulated as a multi-objective optimization problem. The mathematical model is shown as following:

$$maximize \qquad U_k, \forall k \tag{4.5}$$

$$s.t. \qquad R_{l,p} \times X^T_{l,p} \leq (c_1, c_2, ..., c_l) \tag{4.6}$$

$$\sum_{j=1}^{p_i} x_{ij} \geq B^k_{min}, \forall i, k, i \in pair(k) \tag{4.7}$$

$$0 \leq x_{ij} \leq c_l, \forall i, j \tag{4.8}$$

Inequality (4.6) shows the bandwidth allocation is subject to the constraint of the bandwidth capacity of physical links. Inequality (4.7) guarantees that the bandwidth allocate to each VM-VM pair meet the minimum requirement of the application which the pair belongs to. Inequality (4.8) is the boundary of bandwidth which can be allocated.

5. Solution

Centralized bandwidth allocation is subject to NP-hard problem.[13] It is hard to find the solution in polynomial time. The usage of a heuristic algorithm ensures an acceptable runtime of bandwidth allocation problem since it reduces the complexity of the search space significantly. Within heuristic approaches, the elitist Non-Dominated Sorted Genetic Algorithm-II (NSGA-II)[11] provides a good trade-off between runtime and quality of the result. Having benefited from NSGA-II, we find Pareto optimal solutions for the bandwidth allocation problem. Furthermore,

we propose a multiple attribute decision approach to refine a best solution from the Pareto solution set.

5.1. *NSGA-II based multi-objective optimization*

NSGA-II was proposed by Deb *et al.* in 2002, which solves multi-objective optimization problem by using the concept of domination. The initialed population is sorted into fronts. The first front is non-dominant set in the current population and the individuals in the second front are dominated by individuals in the first front and so on. Each individual in the front is assigned rank (fitness) values based on the front in which they belong to. Individuals in the first front are given a rank value of 1 and individuals in the second front are assigned the rank value of 2 and so on. In addition, crowding distance, a parameter that measures how close an individual is to its neighbors, is also calculated for each individual. A larger average crowding distance indicates a greater diversity in the population. The time complexity of NGSA-II is $O(mN^2)$, where m is the number of objectives, and N is the size of population.

The steps involved in NSGA-II are as follows:

(i) *Population Initialization*: The number of individuals in the population and the number of generations is determined beforehand. Specifically, the initiated population is generated using uniform distribution according to the constraint (4.8).
(ii) *Non-Dominated Sort*: Sorting the population into fronts based on non-domination relationship, each individual is assigned a rank value.
(iii) *Crowding Distance*: Once the sorting is completed, crowding distance is assigned. Crowing distance is calculated by using the Euclidian distance between each individual in a front. Hence the individuals within the same rank can be selected based on the value of crowding distance.
(iv) *Selection*: Once the individuals are sorted based on rank and crowding distance. A binary tournament selection is carried out. An individual with a lower rank and a higher crowing distance is more likely to be selected.
(v) *Crossover*: Intermediate crossover is used in our solution with a crossover fraction equal to 2 by the number of total paths. That is, p.
(vi) *Mutation*: To make population diverse, Gaussian mutation is utilized within a mutation probability 2 by the number of total paths too.
(vii) *Recombination and Selection*: The offspring population is combined with the parent population and selection is performed to generate the new generation. Since all the previous and current best individuals are added in the population, elitism is ensured.
(viii) Steps (iv)–(vii) are repeated for the number of generations defined in step (i).

5.2. *Weight-based multi-attribute decision approach*

NSGA-II gives a Pareto set of optimal solutions, in other words, the front of the first rank may have several individuals. The values of different objectives vary from

individuals which make a difficulty for choosing the best solution. In this situation, the crowding distance-based sorting makes no contribution to our problem. Because this method does not consider the value of utility parameters. To solve this problem, we propose a weight-based multi-attribute decision approach to refine the solutions. The decision matrix can be generated as

$$D = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \dots & x_{m,n} \end{pmatrix} \tag{5.1}$$

The m rows denote the individuals in the first rank front, and the n columns denote the applications. The element x_{ij} in the matrix is the utility of application j under the i_{th} solution.

To be mentioned, we provide an interface to tenants where the tenants can submit their network performance requirements by specifying B_{min}, α, β. What behind the interface is that to get higher quality of service, the tenant always appear for larger α and/or β, which means a higher payment. From the perspective of the cloud providers, it is necessary to meet the requirements of tenants who pay more first. In consideration of this, the weight is naturally calculated by the throughput-sensitive and delay-sensitive coefficient

$$\lambda_i = \sqrt{\alpha_i^2 + \beta_i^2} \tag{5.2}$$

λ_i being small implies that both coefficients are small and being large means that at least one coefficient is large.

Then the weight be normalized as

$$w_i = \frac{\lambda_i}{\sum_{i=1}^{n} \lambda_i} \tag{5.3}$$

Once the weight is assigned, the evaluation of each solution in the first rank front can be given by

$$A_i = \sum_{j=1}^{n} w_j U_j(x_{i1}, x_{i2}, ..., x_{in}) \tag{5.4}$$

The preference ordering of the solutions is sorted in descending order according to the evaluation value A_i. Solution with the largest evaluation value will be chosen as the best solution.

It should be noted that the weight we used here is quite different from what we described in related work. The nature of our bandwidth guarantees is based on the utility of application. Weight is only used for refining the solutions. And the weight we used here is a combination of utility parameters rather than a single value.

5.3. *Implementation discussion*

Our mechanism provides differentiated bandwidth guarantees for cloud data centers by implementing bandwidth allocation in the virtualization layer of the physical host. After receiving the optimal solution from the centralized management unit, virtual switch automatically sets a bandwidth capping to each VM-VM pair with

$$up_i = \sum_{j=1}^{p_i} x_{ij} \tag{5.5}$$

Another thing we need to do is to allocate the bandwidth x_{ij} to each corresponding path j. We make a simple modification on currently commonly used Equal Cost Multiple Path (ECMP) routing protocol in data centers by adding a weight to each path. We can distribute packets to each path according to $X_i = (x_{i1}, x_{i2}, ..., x_{ip_i})$. The weight can be calculated by

$$w_{ij} = \frac{x_{ij}}{\sum_{k=1}^{p_i} x_{ik}} \tag{5.6}$$

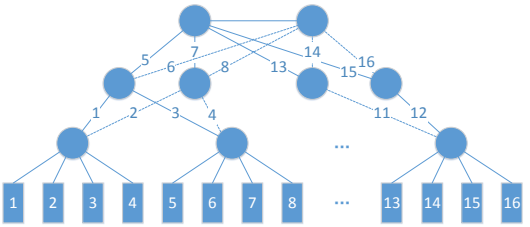
6. Simulations

We conduct extensive simulations to show that the proposed differentiated bandwidth guarantee mechanism achieves three network sharing objectives. All experiments are performed on a server with 16G memory and 3.3GHz AMD FX-6100 Six-Core Processor using *Matlab*. We use three typical structures: tree,[3] fat-tree,[23] and VL2,[12] which represent data center networks of different architectures. Small scale instances are shown in Figure 2.

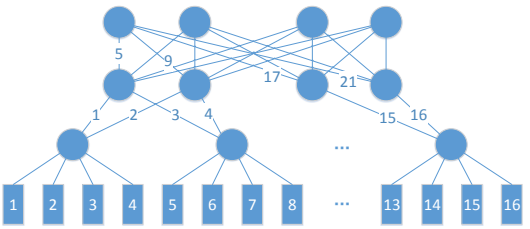
Tree structure is commonly used in enterprise data center, which is built from high-cost hardware. Due to the cost of the equipment, the capacity between different branches of the tree is typically oversubscribed by factors of 1:5 or more, which limits the communications between servers/virtual machines. Fat-tree and VL2 are designed for cloud-oriented data centers built from commodity switches, providing extensive path diversity between servers. Compared to Fat-tree, VL2 has even more redundant paths for each VM-VM pairs. We mark each links according to the approach we proposed in Section 3, the number is shown on each physical links.

In all simulations, bandwidth capacity of each link is set to be 1000 Mbps. We simulate a group of VM-VM pairs running in all network topologies. The scales range from 10 to 1000 VM-VM pairs. The throughput sensitive and delay sensitive coefficients of VM-VM pairs are set to be uniform random within [1, 10], the minimum bandwidth requirements are set to be random within [1, 10] Mbps. In each experiment, we set parameters in the NSGA-II algorithm with the size of population to be 50 and the maximum generation to be 200.

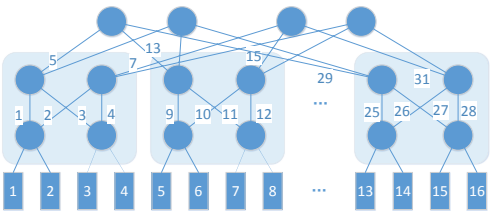
Figure 3 shows the bandwidth allocation time for the three structures. It can be seen that for a data center running 100 VM-VM pairs in tree and fat-tree topologies, we only need several hundred seconds to give the bandwidth guarantees. Bandwidth



(a) Tree



(b) VL2



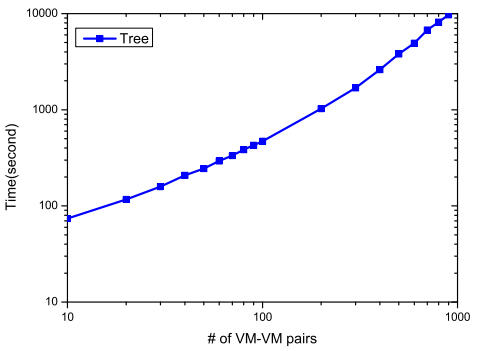
(c) Fat-tree

Fig. 2. Three typical data center architectures.

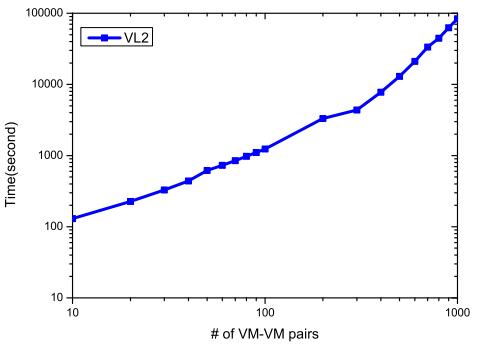
allocation time in VL2 is an order of magnitude higher, because the redundant paths are doubled in this topology. Taking communication between VM 1 to VM 16 as an example, there are 8, 16, 4 paths in tree, VL2 and fat-free topology respectively. For a data center with 1000 VM-VM pairs in fat-tree, we can perform allocation in 102 minutes. We admit the algorithm can be optimized and more effective algorithms can be designed to improve the time complexity. Nevertheless, it should be noted that it is sufficient for our offline bandwidth allocation. The result shows that allocation time only grows quadratic with the number of VM-VM pairs, which shows the scalability of our allocation algorithm.

As the solution meets all constraints we defined in the formulation and the minimum bandwidth guarantee is one of the constraints (constraint (4.7)). Hence the minimum bandwidth requirements of applications are naturally met in our mechanism. To show fine-grained service differentiation, we conduct experiments in three structures with 100 different types of VM-VM pairs, each pair represent one type of application. The results are shown in Figure 4.

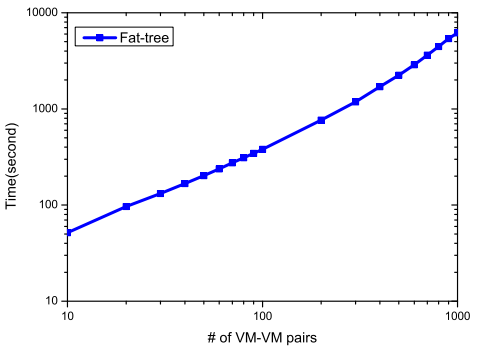
It is shown that in all topologies, when we fix throughput sensitive coefficient α, the utility that application obtained increases with the growing of delay sensitive coefficient β. However, when we fix delay sensitive coefficient β, the utility appears



(a) Tree



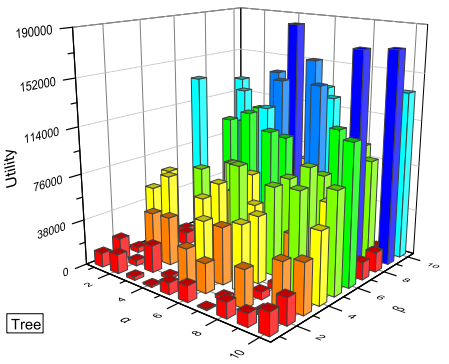
(b) VL2



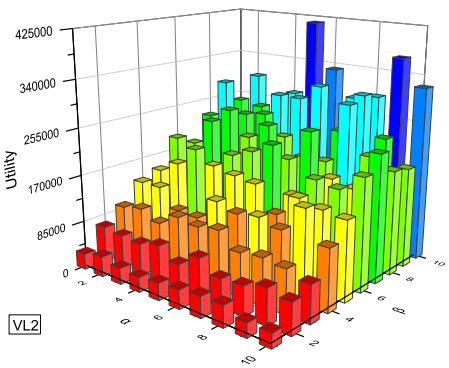
(c) Fat-tree

Fig. 3. Bandwidth allocation time.

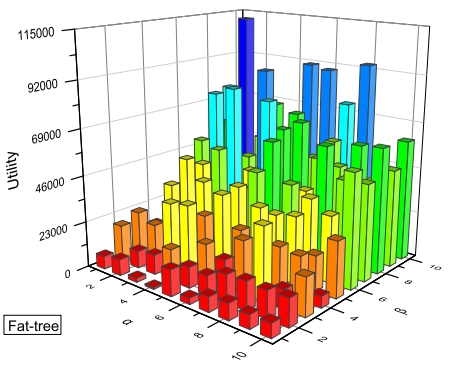
to be fluctuate. This phenomenon most probably resulted from two reasons: (1) the minimum bandwidth requirements vary from different applications, value of minimum bandwidth requirement determines the basic utility of throughput sensitive



(a) Tree



(b) VL2



(c) Fat-tree

Fig. 4. Utilities of different applications.

application. (2) VMs belong to throughput sensitive applications may be placed in the same physical server. Therefore, the utility introduced by throughput is constrained by the capacity of access links. The inherent reason behind these two reasons is: the placement of virtual machines/VM-VM pairs affect the result. Taking VM

Table 2. Bandwidth allocation result.

α	β	$Path_1$	$Path_2$	$Path_3$	$Path_4$	$Bandwidth$	$Utility$
1	1	619	389	507	105	1622	6489
1	2	415	418	492	679	2004	8026
2	2	0	423	262	383	1068	8558
3	6	209	589	135	170	1103	13255
4	1	89	109	186	426	810	13003
5	5	341	598	169	876	1985	39716
6	4	579	138	31	107	855	20570
6	8	491	557	14	546	1608	20570
8	10	209	389	983	360	1941	62172
10	10	866	468	76	98	1510	60410

positions into consideration to share the data center network is a direction of our future research.

In Figure 4(a), the number of physical links (16 in our simulations) is the least among three structures. Due to its low capacity, the bandwidth guarantee is a little weak, application with $\alpha = 10, \beta = 8$ do not get deserved guarantee. This condition is improved with the use of cloud-oriented topologies fat-tree and VL2.

We tried to compare our algorithm with several related algorithms. But those algorithms do not consider application performance affected by delay. They only allocate the size of bandwidth according to the weight of application, regardless of paths which are used by applications. In Table 2, we randomly capture bandwidth allocation for 10 applications in fat-tree.

The result shows the fact that network performance is not in proportion to the bandwidth that an application got. For instance, application with $\alpha = 1, \beta = 2$ obtained the largest bandwidth among other applications, but its utility is lower than other applications in most cases. The reason is that others use paths with less delay, making the utility of these applications to be improved remarkably. Our proposed bandwidth guarantee mechanism not only allocates bandwidth in different sizes, but also considers the path delay which affects the network performance of applications. Hence it outperforms proposed weighted-based approaches.

7. Conclusions

In this paper, we studied network sharing problem in cloud data centers. Aiming to provide fine-grained differentiated bandwidth guarantees, we proposed an application utility based interface to tenants whereby they can express application performance requirements to cloud providers. We identified three key objectives that data center network should satisfy: *minimum bandwidth guarantees*, *fine-grained service differentiation* and *adaptability*. Extensive simulations verified that our proposed mechanism achieves above goals. The mechanism benefits both tenants and cloud providers. Besides, it opens the door for designing differentiated bandwidth pricing model and the on-demand access to network resource offered by cloud data centers

becomes possible. In the future, we are going to deploy a small scale testbed of data center, and validate the performance of our proposed approach on that testbed.

Acknowledgments

This work is supported by the National Natural Science Foundation of China(No. 61002011); the Fundamental Research Funds for the Central Universities(No. 2013RC1104); the 863 Program of China(No.s 2013AA013303); the Open Fund of the State Key Laboratory of Software Development Environment(No. SKLSDE-2009KF-2-08), Beijing University of Aeronautics and Astronautics.

References

1. Amazon elastic compute cloud (amazon ec2). URL http://aws.amazon.com/cn/ec2/#pricing.
2. Open vswitch. URL http://openvswitch.org/.
3. *Cisco Data Center Infrastructure 2.5 Design Guide*. Cisco Systems, Inc., 2007.
4. Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, August 2008.
5. M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.
6. H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards predictable datacenter networks. In *ACM SIGCOMM*, 2011.
7. H. Ballani, D. Gunawardena, and T. Karagiannis. Network sharing in multi-tenant datacenters. Technical report, Microsoft Research, Cambridge, 2012.
8. P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 164–177. ACM, 2003.
9. T. Benson, A. Anand, A. Akella, and M. Zhang. Understanding data center traffic characteristics. *ACM SIGCOMM Computer Communication Review*, 40(1):92–99, 2010.
10. B. Braden, D. Clark, and S. Shenker. Integrated service in the internet architecture: an overview, 1994.
11. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.
12. A. Greenberg, J.R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D.A. Maltz, P. Patel, and S. Sengupta. Vl2: a scalable and flexible data center network. *Communications of the ACM*, 54(3):95–104, 2011.
13. C. Guo, G. Lu, H.J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. Secondnet: a data center network virtualization architecture with bandwidth guarantees. In *Co-NEXT '10 Proceedings of the 6th International Conference*, pages 1–12, Philadelphia, Pennsylvania, 2010. ACM, ACM. 1921188.
14. U. Javed, M. Suchara, J. He, and J. Rexford. Multipath protocol for delay-sensitive

traffic. In *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*, pages 1–8. IEEE, 2009.

15. V. Jeyakumar, M. Alizadeh, D. Mazieres, B. Prabhakar, C. Kim, and W. Azure. Eyeq: practical network performance isolation for the multi-tenant cloud. In *Proceedings of the 4th USENIX conference on Hot Topics in Cloud Ccomputing*, pages 8–8. USENIX Association, USENIX Association, 2012.
16. K. Chen, C. Guo, H. Wu, J. Yuan, Z. Feng, Y. Chen, S. Lu, and W. Wu. Dac: Generic and automatic address configuration for data center networks. *Networking, IEEE/ACM Transactions on*, 20(1):84–99, 2012.
17. S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The nature of data center traffic: measurements & analysis. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 202–208, Chicago, Illinois, USA, 2009. ACM.
18. A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. kvm: the linux virtual machine monitor. In *Linux Symposium*, page 225, 2007.
19. Y.A. Korilis, A.A. Lazar, and A. Orda. Architecting noncooperative networks. *Selected Areas in Communications, IEEE Journal on*, 13(7):1241–1251, 1995.
20. Y.A. Korilis, A.A. Lazar, and A. Orda. Capacity allocation under noncooperative routing. *Automatic Control, IEEE Transactions on*, 42(3):309–325, 1997.
21. T. Lam and G. Varghese. Netshare: Virtualizing bandwidth within the cloud. Technical report, UCSD Technical Report, 2009.
22. V.T. Lam, S. Radhakrishnan, R. Pan, A. Vahdat, and G. Varghese. Netshare and stochastic netshare: predictable bandwidth allocation for data centers. *SIGCOMM Comput. Commun. Rev.*, 42(3):5–11, 2012. 2317309.
23. R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. Portland: a scalable fault-tolerant layer 2 data center network fabric. In *ACM SIGCOMM Computer Communication Review*, volume 39, pages 39–50. ACM, 2009.
24. A. Orda, R. Rom, and N. Shimkin. Competitive routing in multiuser communication networks. *IEEE/ACM Transactions on Networking (ToN)*, 1(5):510–521, 1993.
25. L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica. Faircloud: Sharing the network in cloud computing. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 187–198. ACM, 2012.
26. H. Rodrigues, J.R. Santos, Y. Turner, P. Soares, and D. Guedes. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. In *Proceedings of the 3rd conference on I/O virtualization*, 2011.
27. A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha. Sharing the data center network. In *USENIX NSDI*, volume 11, 2011.