Achieving Bandwidth Guarantees in Multi-Tenant Cloud Networks Using a Dual-Hose Model

Meng Shen*, Lixin Gao†, Ke Xu‡, and Liehuang Zhu*
*School of Computer Science, Beijing Institute of Technology, P. R. China
†Department of Electrical and Computer Engineering, University of Massachusetts Amherst, USA
‡Department of Computer Science, Tsinghua University, P. R. China
{shenmeng, liehuangz}@bit.edu.cn; lgao@ecs.umass.edu; xuke@tsinghua.edu.cn

***Abstract*—In public cloud networks, applications of different tenants compete for the shared network bandwidth and thus might suffer from unpredictable performance. It is desirable for cloud providers to offer tenants with bandwidth guarantees. However, it is challenging to precisely abstract tenant bandwidth requirements for their intra- and inter-tenant communications and to achieve work conservation simultaneously.**

In this paper, we first propose a dual-hose model, a novel tenant requirement abstraction that decouples bandwidth guarantees for a tenant's inter-tenant communications from those for its intra-tenant communications. We then develop a new VM placement algorithm to optimize operational goals of cloud providers, while providing tenants with minimum bandwidth guarantees captured by the dual-hose model. Finally, we design a dynamic bandwidth allocation strategy to achieve work conservation. Through extensive simulation results, we show that our solution provides bandwidth guarantees for tenant requests while improving the overall request throughput by 5.3%.

I. Introduction

Cloud computing provides enterprises with an opportunity to migrate their services and applications to public cloud platforms. Many of today's public cloud providers, such as Amazon EC2 [1], only offer isolation of computing resources (*e.g.*, CPU and memory), without any type of guarantees on network resources. As a result, tenants compete for the scarce bandwidth resources and experience unpredictable completion time [3, 13]. It is highly desirable for cloud providers to provide bandwidth guarantees such that tenants can predict the worst-case performance of their applications.

Many recent proposals tackle this problem, including reservations [2–4, 7–9, 11], and fair-sharing [5, 6]. These proposals focus only on intra-tenant communications. According to measurement results on real traces in datacenters [10, 14], the communication between tenants (*i.e.*, inter-tenant traffic) is prevalent and accounts for 10%-30% of the total datacenter traffic. Recently, Ballani *et al.*[10] propose a hierarchical hose model with communication dependency. In this model, each VM of a tenant gets a minimum bandwidth guarantee irrespective of intra- or inter-tenant traffic and the whole tenant gets a bandwidth guarantee for its aggregate inter-tenant communication with tenants it depends on. However, mixing guarantees for two types of traffic is problematic as we shall illustrate with examples in Sec. II.

There are two challenges in simultaneously providing minimum bandwidth guarantees for a tenant's intra- and inter-tenant traffic. *First*, it is sophisticated to abstract bandwidth requirements for a tenant without knowing its detailed traffic pattern. For instance, the hierarchical hose model [10] introduces an aggregate abstraction for a tenant's inter-tenant traffic. Then each physical machine hosting the tenant's VMs needs to reserve an amount of the aggregate bandwidth for inter-tenant traffic. Therefore, such an abstraction results in bandwidth over-provisioning and thereby reduces the number of concurrent tenants in datacenters. The *second* challenge is how to share the network in order to achieve work conservation. Since network bandwidth is usually significantly larger than per-tenant guarantee (*e.g.*, tens of Gpbs vs. hundreds of Mbps), work-conservation enables tenants to obtain more bandwidth and thus to reduce their job completion time. However, tenant traffic is prone to interfere with each other. Thus, cloud providers also need to ensure the fairness in network sharing when pursuing work conservation.

In this paper, we propose a new bandwidth requirement abstraction named the dual-hose model, which decouples the guarantees for a tenant's inter-tenant traffic from those for its intra-tenant traffic. Through decoupling, the dual-hose model can precisely capture bandwidth requirements for tenants, without introducing the over-provisioning problem. Our major contributions are three-fold.

First, we design the dual-hose model to abstract tenant bandwidth requirements and use examples to briefly illustrate its benefits for cloud providers and tenants (Sec. II).

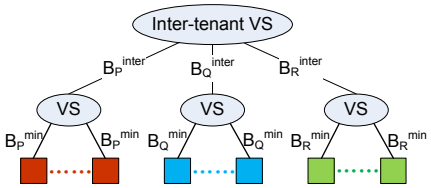
Second, based on the dual-hose model, we formulate the VM placement as an integer programming problem and devise an efficient VM placement algorithm (Sec. III). Then we develop a new bandwidth allocation strategy that achieves work-conservation and avoids the interference between guarantees for intra- and inter-tenant flows on each link (Sec. IV).

Finally, through extensive simulation results, we show that the dual-hose model, combined with the VM placement and the bandwidth allocation strategy, can improve the request throughput by 5.3%, and reduce the average request completion time by 9.1% (Sec. V).

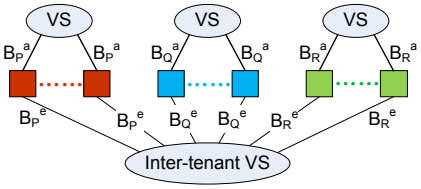
This work has been supported in part by NSF Grants (CNS-1217284 and CCF-1018114), NSFC Project (61170292, 61472212, 61161140454), National Science and Technology Major Project (2012ZX03005001), 973 Project of China (2012CB315803), 863 Project of China (2013AA013302) and EU Marie Curie Actions EVANS (PIRSES-GA-2010-269323).

Meng Shen and Ke Xu are co-corresponding authors.

978-1-4799-7575-4/14/$31.00 ©2014 IEEE



(a) A hierarchical hose model [10]



(b) A dual-hose model

Fig. 1. Characterizing the minimum bandwidth guarantees for tenants. VMs of different tenants are marked with different colors.

II. Dual-Hose Model

A. Tenant Request Abstractions

A hierarchical hose model is designed in [10] to provide guarantees for both inter- and intra-tenant traffic. With this model, a tenant request comprises four parameters:

- N, the total number of VMs the tenant requests.
- B^{min}, the minimum bandwidth guarantee for each VM of the tenant, irrespective of whether the traffic is destined to the same tenant or not.
- B^{inter}, the minimum bandwidth guarantee for the tenant's aggregate inter-tenant traffic.
- S, a set of tenants that this tenant expects to communicates with. Specially, $S = \{*\}$ means this tenant can communicate with any other tenants.

Fig. 1(a) shows such a model, where VMs for each tenant are connected to a dedicated virtual switch (VS) by links whose capacities are equal to VMs' minimum bandwidth guarantees. In datacenters, the locality of VMs for the same tenant depends on many factors, such as VM slot fragmentation, available link bandwidth and fault-torrent consideration [14], and hence, varies a lot. The aggregate inter-tenant guarantee implicitly requires cloud providers to provide that amount of bandwidth on the access link of each server hosts VMs for the tenant. Therefore, this approach results in bandwidth over-provisioning in accommodating tenant requests.

We propose a *dual-hose* model, which is a new tenant requirement abstraction that decouples the guarantee for each VM's inter-tenant traffic from the one for its intra-tenant traffic. Fig. 1(b) shows such a model. A tenant requesting N VMs is also characterized by a four-tuple $\langle N, B^a, B^e, S\rangle$. It means that each VM is provided with a minimum bandwidth guarantee B^a for its traffic to other VMs of the same tenant and a minimum bandwidth guarantee B^e for its traffic to other tenants. Hereafter we use superscripts a and e to indicate intra- and inter-tenant traffic, respectively.

To make the two abstractions comparable, we define

$$B^e = \frac{B^{inter}}{N}; \quad B^a = B^{min} - B^e \tag{1}$$

Equation (1) means that, if we evenly split the aggregate inter-tenant guarantee B^{inter} in the hierarchical hose model over N VMs, each VM gets on average an inter-tenant guarantee B^e. Since $B^{inter} < N * B^{min}$ usually holds [10], the intra-tenant guarantee of each VM B^a is positive, *i.e.*, $B^a = B^{min} - \frac{B^{inter}}{N} > 0$. Therefore, with the dual-hose model, each VM of the tenant also gets a total amount of B^{min} bandwidth guarantee, while the entire tenant gets an aggregate B^{inter} bandwidth guarantee for its inter-tenant traffic.

TABLE I. Parameter Settings for two requests in Example 1

Requests	Hierarchical hose model				Dual-hose model			
	N	B^{min}	B^{inter}	S	N	B^a	B^e	S
Tenant P	2	1000	400	$\{*\}$	2	800	200	$\{*\}$
Tenant Q	6	400	600	$\{P\}$	6	300	100	$\{P\}$

We argue that the decoupling is reasonable for tenants in today's datacenters. First, if we refer to public services offered by cloud providers or third-parties in the marketplace as individual tenants, each VM for a tenant that relies on such services is likely to communicate with VMs for service tenants, which results in inter-tenant traffic. For example, all VMs requiring persistent storage need to communicate with tenants that provide storage services [10]. Second, load-balancing techniques are commonly applied to production applications to distribute their load, including traffic load, over their VMs. Therefore, the dual-hose model assumes homogenous inter-tenant guarantees for VMs in the same tenant. However, it can be easily extended to heterogenous cases by assigning VMs different weights.

B. Benefits of Applying the Dual-Hose Model

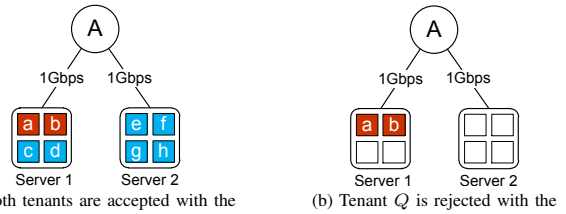
Decoupling two types of guarantees is beneficial to both cloud providers and tenants, which is elaborated as follows.

1) The dual-hose model enables cloud providers to accommodate more tenants and thus increase their revenues.

Example 1: We use an example scenario with two tenant requests for explanation, where tenants P and Q arrive in sequence. According to Equation (1), the parameter settings for the two requests are summarized in Table I. The physical topology comprises two severs connected to a switch by 1 Gbps links, as shown in Fig. 2(a). Each server can host 4 VMs. We assume that when tenant P arrives, its 2 VMs are placed in Server 1 (*i.e.*, VMs in red). For tenant Q, there is only 1 possible placement, *i.e.*, accommodating 2 VMs in Server 1 and 4 VMs in Server 2. Now we should check the feasibility of this placement.

With the dual-hose model, the bandwidth requirement on the physical link for this placement comprises guarantees for three types of traffic, including intra-tenant traffic for tenants P and Q, and inter-tenant traffic between the two tenants. As shown in Equation (2), the bandwidth requirement is exactly 1000 Mbps and thus both tenants can be accepted.

$$\min(2B_P^a, 0) + \min(2B_Q^a, 4B_Q^a) + \min(2B_P^e, 4B_Q^e) \tag{2}$$



(a) Both tenants are accepted with the dual-hose model

(b) Tenant Q is rejected with the hierarchical hose model

Fig. 2. An example scenario illustrating the benefit of the dual-hose model. Red boxes indicate VMs allocated to tenant P and Blue for tenant Q.

With the hierarchical hose model, since the guarantee for a tenant's inter-tenant traffic is already included in the total guarantee for its VMs, the calculation in Equation (2) is invalid. The bandwidth requirement on this link is the minimum upstream requirement of the two servers, *i.e.*, $\min(B_P^{inter}+2B_Q^{min}, 4B_Q^{min})$=1200 Mbps. So this placement is infeasible and tenant Q is rejected as shown in Fig. 2(b).

The reason lies in that from the perspective of individual VMs, the hierarchical model does not differentiate guarantees for different types of traffic. In order to provide the aggregate inter-tenant guarantee, it requires more bandwidth on physical links than the dual-hose model. (See the proof in [18]).

In the above example, if we choose a different placement for tenant P (*i.e.*, hosting 1 VM by each server), then tenant Q is rejected no matter which model is used. Because the bandwidth requirement for accepting two tenants is 2100 Mbps and 2200 Mbps for the dual-hose model and the hierarchical hose mode, respectively. Therefore, given a batch of fixed datacenter resources (*i.e.*, server slots and link bandwidth), an efficient online VM placement is critical for accepting more future requests. We will focus on this issue in Sec. III.

2) The dual-hose model prevents two types of guarantees from interfering with each other.

Example 2: We decrease the guarantees required by tenant Q in Table I such that both tenants can be simultaneously accommodated with the hierarchical hose model. The modified request of tenant Q is denoted as $B_Q^{min} = 300$ Mbps and $B_Q^{inter} = 600$ Mbps. Then the feasible placement is the same as shown in Fig. 2(a). Hence, the bandwidth guarantees for intra- and inter-tenant flows on link l connecting two servers are 600 and 400 Mbps, respectively. To provide the required bandwidth guarantees for both tenants, the actual bandwidth allocated to two types of flows on a link should be proportional to their bandwidth guarantees and *independent* of the number of flows each type has.

Suppose there are only four flows traversing link l, namely $a \to e, b \to e, c \to f$ and $d \to f$. In this paper *flow* refers to all traffic between the same pair of VMs. In [10], a bandwidth allocation strategy is proposed to work with the hierarchical hose model. Each flow is assigned a weight for bandwidth sharing. The weight for a flow between two different tenants depends on their respective number of inter-tenant flows, *e.g.*, $w_{a,e} = w_{b,e} = \min(\frac{B_P^{inter}}{2}, \frac{B_Q^{inter}}{2}) = 200$, which takes the minimum per-flow inter-tenant guarantee from the source and destination tenants. The weight for a flow between two VMs within the same tenant depends on their respective number of flows, *e.g.*, $w_{c,f} = w_{d,f} = \min(\frac{B_Q^{min}}{1}, \frac{B_Q^{min}}{2}) = 150$. The bandwidth on link l allocated to each flow is determined by the ratio of its weight over the total weight of all flows traversing the link. The work-conserving requirement entails that the link capacity (1000 Mbps) should be fully utilized by the four flows. Therefore, $B_{a,e} = B_{b,e} = \frac{2000}{7}$ Mbps and $B_{c,f} = B_{d,f} = \frac{1500}{7}$ Mbps. The actual bandwidth on link l allocated to intra- and inter-tenant flows are $\frac{3000}{7}$ and $\frac{4000}{7}$ respectively, which are not in proportion to their bandwidth guarantees.

The fundamental cause of the problem is that the current bandwidth allocation strategy determines the share for intra- and inter-tenant flows based only on their weights. Hence, the bandwidth share for each type of flows depends on the flow-level communication pattern. In Sec. IV, we propose a new bandwidth allocation strategy to ensure the bandwidth acquired by each type of flows.

III. VM Placement

Based on the dual-hose model, we develop a solution to provide tenants with bandwidth guarantees for both intra- and inter-tenant traffic. It mainly comprises two components: *VM placement* that maps VMs of a newly arrived tenant request onto physical servers, and *bandwidth allocation* that periodically updates the bandwidth allocated to VM-to-VM flows so as to achieve work conservation. Here we focus on tree-like datacenter topologies.

A. Optimal Placement Problem

Given a tenant request, the VM placement is to decide how to accommodate its VMs in physical serves while satisfying the following constraints:

- Bandwidth constraints. The bandwidth requirement on each link to provide VMs with required guarantees should not exceed residual link capacity.
- Slot constraints. The number of VMs each physical server hosts should not exceed its available slots.

Since there might be multiple feasible placements satisfying the constraints, a provider needs to find a single *best* placement from the possibilities. According to diverse operational goals, the best placement can be defined in different ways, among which we list two representative ones [4].

- *Maximizing available bandwidth*. Since bandwidth at a higher level in the tree-like topology is scarcer, a typical goal is to select a placement that maximizes the available bandwidth of higher-level links, which in turn helps to accept more future tenants.
- *Maximizing the square sum of empty slots*. The available slots in a partially utilized subtree at each level become slot fragments (similar to those in file systems). A placement utilizing smaller fragments is preferable, because larger fragments can be left for further requests.

In this paper, we jointly consider the two goals when finding the best placement. Given a feasible VM placement V, we define Λ_{bw} as the cost function of bandwidth under V and Λ_{slot} as the cost function of physical slots, respectively. Based on the two functions, we also define a global cost

TABLE II. NOTATIONS FOR VM PLACEMENT

Notation	Definition
m_i	The number of available slots in server i
v_i	The number of the tenant's VMs accommodated in server i
V	The placement of the current tenant request, $V = \{v_i\}$
$\Phi(V)$	The cost function of a specific placement V
Λ_{bw}	The cost function of bandwidth consumption under V
Λ_{slot}	The cost function of physical slot consumption under V
ω_j	The weight of link j
f_j	The incremental bandwidth requirement on link j under V
D_j	The server set in the subtree under link j
r_j	The residual capacity of link j

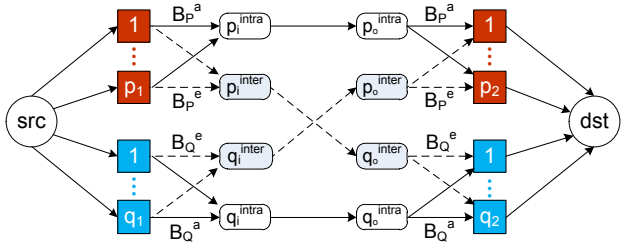


Fig. 3. A communication network with the dual-hose model. VMs for different tenants are marked with different colors.

function of the current placement $\Phi(V) := \langle \Lambda_{bw}, \Lambda_{slot} \rangle$, as well as an ordering that allows us to compare values Φ_1 and Φ_2 determined by two different placements. Since bandwidth resources, particularly at higher levels, are scarcer than physical slots, the ordering we choose is *lexicographic* [17], namely, $\langle a_1, b_1 \rangle > \langle a_2, b_2 \rangle$, if and only if $a_1 > a_2$, or $a_1 = a_2$ and $b_1 > b_2$. Notations are summarized in Table II.

Among all feasible placements, we search for the best placement V that minimizes the global cost function $\Phi(V)$

$$\text{minimize} \quad \Phi(V) := \langle \Lambda_{bw}, \Lambda_{slot} \rangle \tag{3a}$$

$$\text{subject to} \quad \Lambda_{bw} = \sum\nolimits_j \omega_j f_j \tag{3b}$$

$$f_j = \Delta bandwidth_j(\sum_{i \in D_j} v_i), \forall j \tag{3c}$$

$$\Lambda_{slot} = 1/(\sum_i (m_i - v_i)^2) \tag{3d}$$

$$\sum\nolimits_i v_i = N \tag{3e}$$

$$v_i \le m_i, f_j \le r_j, \forall i, j \tag{3f}$$

Remarks:

Constraint sets in Equations (3b)-(3c) define Λ_{bw} by summing up the bandwidth cost over all links, where $\Delta bandwidth_j(n)$ denotes the incremental bandwidth requirement on link j when placing n VMs in the subtree under link j. We can force the placement to use as less higher-level link bandwidth as possible by assigning them higher weights, ω_j.

Constraint set in Equation (3d) defines the cost of physical slot consumption. Note that we maximize the square sum of empty slots in servers by minimizing the cost function Λ_{slot}.

Constraint sets in Equations (3e)-(3f) ensure that all N VMs in the tenant's request are accommodated in physical servers, while satisfying the bandwidth and slot constraints.

B. Characterizing Bandwidth Requirements

From Equation (3), we find that characterizing bandwidth requirements is crucial to calculate the objective function and to verify the bandwidth constraints. This problem has been well studied in the context of providing VMs with only intra-tenant bandwidth guarantees [3]. However, it becomes more complicated to simultaneously consider intra- and inter-tenant guarantees, because the bandwidth requirement of a newly-arrived tenant on each link is determined not only by the tenant, but also by the communication dependency on existing tenants.

The bandwidth requirement for the specific placement in Example 1 is captured in Equation (2). Now we also use a two-tenant (P and Q) example to generalize this calculation process. Assume that tenants P and Q are an existing and a new tenant, respectively. In a tree-like topology, each link bridges two disjoint parts: the subtree under the link and the rest of the tree. Hereafter we refer to these two parts as *in* and *out*. For a certain link, without loss of generality, we assume there are p_1 VMs for tenant P in its subtree and p_2 VMs out of its subtree. Now we should calculate the bandwidth requirement on this link for any possible placement (*i.e.*, placing q_1 and q_2 VMs for tenant Q in each part).

To precisely capture the bandwidth requirement of intra- and inter-tenant traffic on each link, we employ a *communication network* similar to the one in [10]. As shown in Fig. 3, a communication network is a directed graph with a source node src and a sink node dst. Each VM in the subtree of the link is represented by a node connected with src, while each VM out of the subtree is represented by a node connected to dst. To distinguish physical links in datacenters from links in the communication network, we refer to the former as *links* and the latter as *edges*.

For each tenant, we create two intra-tenant nodes (*e.g.*, p_i^{intra} and p_o^{intra}) and two inter-tenant nodes (*e.g.*, p_i^{inter} and p_o^{inter}), which can be viewed as the intra- and inter-tenant virtual switches in Fig. 1(b). VM nodes for a tenant are connected simultaneously to its intra-tenant nodes by edges whose capacity is the minimum intra-tenant guarantee for VMs and to corresponding inter-tenant nodes by edges whose capacity is the minimum inter-tenant guarantee. These edges represent the per-VM bandwidth constraints. Two intra-tenant nodes for the same tenant are connected together, while inter-tenant nodes for tenants that have communication dependencies are connected to each other. All edges without explicit capacity labels are associated with an infinite capacity.

The bandwidth requirements of tenants on each link is equivalent to the maxflow of the communication network. Therefore, for any possible placement of the new request, we can construct such a network to obtain its bandwidth requirement on each link.

C. Placement Algorithm

The VM placement problem defined in Equation (3) is reduced to an integer programming problem. Since the cost for searching for all possible placements becomes combinational in large-scale datacenters (*e.g.*, thousands of servers), we develop a heuristic VM placement algorithm (Algorithm 1).

Algorithm 1: VM Placement

Input: Topology tree $T(L)$ with a $root$ link and a request $r :< N, B^a, B^e, dep >$
Output: VM placement of the request r

```
1  for each link l ∈ L do
2      Calculate its feasibility vector FV_l and M_l
   // start from the root link
3  if Place(r, root, N) == N then
4      return true
5  else
6      return false
   // Allocate v VMs for tenant r in the subtree
      under link l
7  Function Place(r, l, v)
8  if (level(l) == 0) then
       // l is an outbound link of a server
9      count ← (FV_l, v)^+
10 else
11     if (level(l) == 1) then
           // l is an outbound link of a rack
12         cld_l ← l's children ordered ascendingly by empty slots
13     else
14         cld_l ← l's children ordered descendingly by max(FV)
15     remain ← v − Σ_{k∈cld_l} M_k
16     for each k ∈ cld_l do
17         v_k ← M_k+ remain;    tmp ← Place(r, k, v_k)
18         if tmp != -1 then
19             remain ← remain − (tmp − M_k)
20             count ← count + tmp
21         else
22             return -1
23 if count ∉ FV_l then
24     return -1
25 return count
```

Assume that there is a *root* link as the outbound link of the entire tree topology. Given a specific request, the basic idea behind this algorithm is to recursively place VMs using function Place in the subtree(s) under each link while satisfying the bandwidth and slot constraints. To achieve the optimization goal in Equation (3a), subtrees are selected with different priorities. First, among subtrees whose outbound links are close to the core network, we try to minimize the bandwidth requirement across the core network by selecting those that can accommodate more VMs (line 14). Second, among subtrees as racks, the ones with fewer empty slots are chosen first to fully utilize the rack-level fragmentation (line 12). The request is accepted if all its VMs are successfully placed, or rejected otherwise (lines 3-6).

As stated in Equation (3f), the number of VMs that can be placed in a subtree is constrained by two constraints. To facilitate the constraint-checking process, we introduce a structure named feasibility vector (FV) for each link, which indicates the number of VMs that can be accommodated in the subtree under this link. For a tenant request with N VMs, each physical link l maintains an FV with $N+1$ bits denoted as FV_l. For each $n(0 \leq n \leq N)$, we construct a communication network and then verify whether n satisfies slot and bandwidth constraints (*i.e.*, $FV_l(n)$=1) or not (*i.e.*, $FV_l(n) = 0$). Now we can use the FV to represent the two-dimensional constraints.

Unlike the case that considers guarantees only for intra-tenant traffic, placing 0 VMs in a subtree is not always a feasible solution, because the requirement for inter-tenant guarantees might violate the bandwidth constraints. Therefore, each link l also records M_l, which denotes the minimum feasible number in FV_l (lines 1-2).

Since links might have the minimum VMs that must be placed in its subtree, the algorithm first satisfies these minimum VM requirements (line 15), and then tries to place the remaining VMs in their children links' subtrees (lines 16-22) with different priorities illustrated above. For each physical server, it accommodates as many VMs as possible (line 9). Therefore, the actual number of VMs it hosts is defined as

$$(FV_l, v)^+ = \max \{i \mid FV_l(i) = 1 \ \& \ i \leq v\} \tag{4}$$

At last, the algorithm verifies whether the aggregated number of VMs placed in a link's children satisfies its FV constraints.

IV. Bandwidth Allocation

After accommodating VMs for a tenant, the cloud needs to decide the bandwidth allocated to each VM-to-VM flow, while providing the minimum bandwidth guarantees required by the dual-hose model. We assume that the VM rate-control components already exist in the datacenter infrastructure. The goal of a bandwidth allocation strategy is two-dimensional:

Efficiency. The *temporal* goal is to achieve high efficiency in terms of network utilization over time. The static bandwidth partitioning [3, 4] is inefficient because the bandwidth allocated to a tenant is always reserved even during idle periods.

Fairness. The *spatial* goal is to guarantee the fair share of flows of all tenants such that: 1) the bandwidth each VM with active flows acquires should be proportional to its required guarantees, and 2) the bandwidth allocated to intra- or inter-tenant flows on each link should be proportional to their respective guarantees.

A. Bandwidth Allocation Policy

To achieve work conservation, the bandwidth for each flow needs to be periodically updated to fully utilize the spare capacity. Here we introduce the time window to control the bandwidth update frequency. The bandwidth allocated for each flow is updated at the beginning of each time window. The following discussion is limited within a single time window.

Each flow is associated with a weight which indicates its portion in sharing bandwidth. Take a flow from VMs p to q for example. In order to avoid VMs acquiring more bandwidth disproportional to their required guarantees, the weight of flow $p \rightarrow q$, $w_{p,q}$, should be constrained by corresponding guarantees of its source and destination VMs. Other notations are summarized in Table III. Accordingly, $w_{p,q}$ is defined as

$$w_{p,q} = \begin{cases} \min(\frac{B^a_{t(p)}}{N^a_p}, \frac{B^a_{t(q)}}{N^a_q}), & t(p) = t(q) \\ \min(\frac{B^e_{t(p)}}{N^e_p}, \frac{B^e_{t(q)}}{N^e_q}), & t(p) \neq t(q) \end{cases} \tag{5}$$

Now we consider how to split the bandwidth C_l of a specific link l over all flows. Based on flows of all tenants on link l, we use z^a_l and z^e_l to indicate whether there are intra- and inter-tenant flows on this link. To avoid two types of flows interfering with each other, the bandwidth allocated to each type of flows on link l is determined based on their total

TABLE III. NOTATIONS FOR BANDWIDTH ALLOCATION

Notation	Definition
$w_{p,q}$	The weight assigned to the flow from VMs p to q
$B_{p,q}$	The bandwidth allocated to the flow from VMs p to q
F_l^a	The total bandwidth guarantee for intra-tenant flows across link l
F_l^e	The total bandwidth guarantee for inter-tenant flows across link l
W_l^a	The total weight for intra-tenant flows across link l
W_l^e	The total weight for inter-tenant flows across link l
$t(p)$	The tenant VM p belongs to
z_l^a	1 if W_l^a is non-zero, or 0 otherwise
z_l^e	1 if W_l^e is non-zero, or 0 otherwise

bandwidth guarantee. Hence, we define the total bandwidth allocated to all intra- or inter-tenant flows as follows:

$$C_l^a = \frac{z_l^a F_l^a}{z_l^a F_l^a + z_l^e F_l^e} C_l,\ C_l^e = \frac{z_l^e F_l^e}{z_l^a F_l^a + z_l^e F_l^e} C_l. \quad (6)$$

where z_l^a and z_l^e cannot be 0 simultaneously; otherwise, the bandwidth of link l is unnecessary to be allocated, because no flows are currently traversing link l.

The bandwidth flow $p \to q$ acquires on link l is as follows

$$B_{p,q} = \begin{cases} \frac{w_{p,q}}{W_l^a} C_l^a, & t(p) = t(q) \\ \frac{w_{p,q}}{W_l^e} C_l^e, & t(p) \neq t(q) \end{cases} \quad (7)$$

Note that the ultimate bandwidth for flow $p \to q$ is the minimum $B_{p,q}$ over all links along its path.

Let's revisit the bandwidth allocation scenario in Example 2. The bandwidth guarantees for intra- and inter-tenant flows on the link connecting two servers are both 400 Mbps, *i.e.*, $F_l^a = F_l^e = 400$ Mbps. According to Equation (5), the weights for intra-tenant flows $c \to f$ and $d \to f$ are the same, *i.e.*, $w_{c,f}$=$w_{d,f}$=min($\frac{200}{1}$, $\frac{200}{2}$) = 100 and $W_l^a = 200$. Similarly, we have $w_{a,e}$=$w_{b,e}$=min($\frac{200}{1}$, $\frac{100}{2}$) = 50 and $W_l^e = 200$. Hence, after applying Equations (6) and (7), the bandwidth allocated to each flow is derived, where $B_{a,e} = B_{b,e} = 250$ Mbps and $B_{c,f}$=$B_{d,f} = 250$ Mbps. Therefore, the actual bandwidth acquired by two types of flows maintains the same ratio between their required bandwidth guarantees. We show a general validation of the proposed strategy in [18].

B. Bandwidth Allocation Algorithm

The bandwidth allocation algorithm (*i.e.*, Algorithm 2) takes the datacenter topology and the flow set as input, and runs at the beginning of each time window. The flow set can be established in two different ways. First, we can use all active flows in the last time window as the flow set for the current time window. Second, any flow should explicitly register at the the central controller before being allocated with bandwidth. Either way is a trade-off between achieving high bandwidth utilization and lowering the massaging overhead. We leave the investigation of time window as future work.

Upon the VM placement for an accepted tenant, bandwidth guarantees for intra- and inter-tenant flows on each link l (*i.e.*, F_l^a and F_l^e) is already determined by the communication network. In order to carry on dynamic bandwidth allocation, each link needs to record four more variables, namely W_l^a, W_l^e, z_l^a, and z_l^e. Based on the input information, the algorithm computes the weight of each flow (line 3) and then updates the total weight on each link along the flow path (lines 4-5).

Algorithm 2: BW Allocation

Input: Topology tree $T(L)$ and a set of all flows
Output: Bandwidth allocated to each flow

```
// Here * denotes a for an intra-tenant flow or
   e for an inter-tenant flow
```

1 Pre-compute the path $\mathcal{P}_{p,q}$ for each VM pair p and q
2 **for** each flow $p \to q$ **do**
3 Set its weight $w_{p,q}$ according to Eq. (5)
4 **for** each link $l \in \mathcal{P}_{p,q}$ **do**
5 Add $w_{p,q}$ to the total weight W_l^* and set z_l^* to 1
6 **for** each flow $p \to q$ **do**
7 $B_{p,q} \leftarrow w_{p,q} * \min_{l \in \mathcal{P}_{p,q}} \{ \frac{C_l^*}{W_l^*} \}$ (see Eq. (7))

TABLE IV. METHODS IN COMPARISON

Methods	Definition
DU-OPT	Dual-hose model + Optimized VM placement
DU-GRD	Dual-hose model + Greedy VM placement
HR-GRD [10]	Hierarchical hose model + Greedy VM placement

Finally, the bandwidth allocated to each flow is determined as the minimum $B_{p,q}$ on links along the flow path (lines 6-7).

Algorithm 2 presents a centralized computation of bandwidth allocated to live flows. But our strategy can also be realized in a distributed manner. To this end, similar to [9, 10], we need slight modifications on commodity switches to record the parameters described above.

V. EVALUATION RESULTS

In this section we develop a simulator to evaluate the proposed VM placement and bandwidth allocation algorithms.

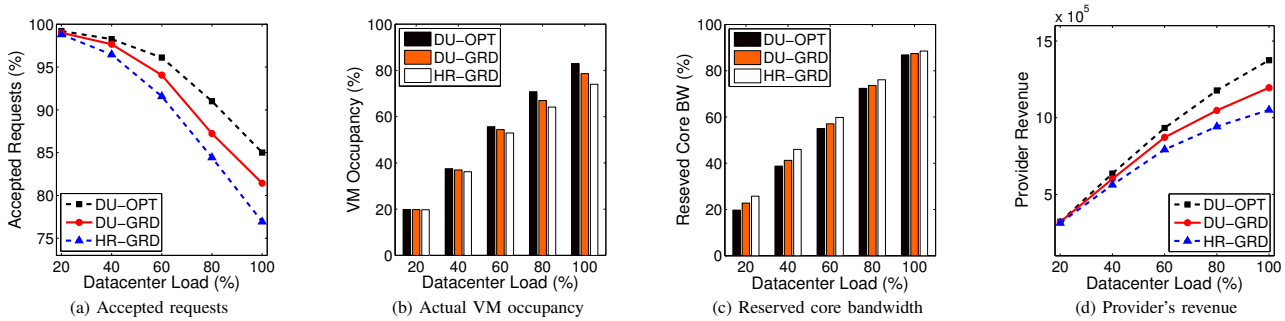
Datacenter Topology. Since a production datacenter usually has thousands of physical servers [10, 14], the datacenter topology in our simulation is a three-level tree topology with 4K servers at level 0 (*i.e.*, leaves). Each server hosts 4 VM slots. Hence, the total number of VMs is 16K. Every 40 servers are connected to a Top-of-Rack (ToR) switch via 1 Gbps links. Similarly, every 10 ToR switches are connected to an aggregation switch and the latter is connected to a single core switch. By configuring capacities of outbound links of switches, we can adjust the oversubscription ratio (default 4).

Tenants. We simulate tenant requests based on workload extracted from recent measurement results [3, 10, 14]. The number of VMs per request is exponentially distributed around a mean of 49. The bandwidth guarantee for intra-tenant traffic (*i.e.*, B^a) is uniformly distributed in $[200, 300]$. We assume 20% of requests have inter-tenant bandwidth guarantees, the amount of which is 20%-40% of their corresponding B^a.

Methods in comparison. We compare the dual-hose model with the hierarchical hose model, the state-of-the-art abstraction [10]. Different combinations are listed in Table IV, where the optimized VM placement refers to Algorithm 1 and the greedy one picks available subtrees from left to right in the tree topology.

A. Effect of VM Placement

In this section, we are dedicated to exploring the effect of different methods in Table IV. To this end, each request is associated with an expected completion time (300 time units on average) and is supposed to complete within that period.



(a) Accepted requests (b) Actual VM occupancy (c) Reserved core bandwidth (d) Provider's revenue

Fig. 5. Comparison of different methods with varied datacenter loads.

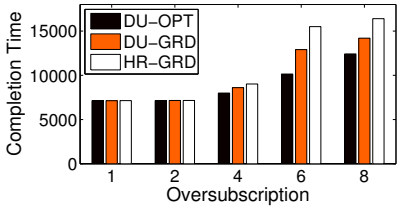


Fig. 4. Completion time with varying oversubscription ratios.

1) Batched Requests: In this scenario, a large number of requests are submitted together and wait to be scheduled to run. This workload captures the *offline* scheduling for time-insensitive requests, where all requests are known in advance. The request scheduling policy tries to maximize the request throughput, by scanning all waiting requests and scheduling any one(s) that can run.

We simulate a stream of 5K requests and plot the total completion time for all requests with different oversubscription ratios in Fig. 4. It shows DU-OPT can save more time compared to the other two methods when the oversubscription ratio is larger, because it reduces the bandwidth requirements in datacenters and thus allows more concurrent requests.

2) Dynamically Arriving Requests: Under this scenario, we assume the request arrival follows a Poisson process with rate λ. If a request cannot be scheduled upon its arrival, it is rejected. This workload captures the online scheduling for dynamic requests, which is common in today's public cloud platforms, such as Amazon EC2. By varying λ, we can control the target datacenter load in terms of VM occupancy denoted by $\lambda \cdot \bar{N} \cdot \bar{T}/M$, where M is the total number of VMs in the datacenter (16K), $\bar{N}$ is the mean request size (49) and $\bar{T}$ is the average completion time (300). We run each simulation for 5000 time units. The maximum number of requests (*i.e.*, 100% load) is also about 5K.

Accepted requests and resource utilization. Fig. 5(a) plots the accepted requests for different methods with varied loads. We observe that under low load (20% and 40%), three methods have similar performance. As the load increases, DU-OPT accepts more requests than its competitors. This result is also confirmed by the VM occupancy in Fig. 5(b), where DU-OPT improves the VM utilization as the load is enlarged.

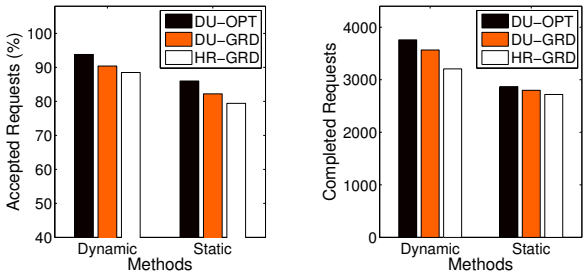
The average reserved bandwidth on core links (*e.g.*, links connected to the core switch) over time is shown in Fig. 5(c). As the target load varies, bandwidth reserved under DU-OPT is less or similar to those under the other two methods, because both the dual-hose model and the optimized VM placement contribute to reducing the bandwidth needs on core links.

Revenue. Today's cloud providers charge tenants a fixed price for each VM per hour without considering the bandwidth [1]. Here we use a simple charging model improved on the VM-only model, *i.e.*, for a tenant request $\langle N, B^a, B^e, S\rangle$ with the expected completion time T, the cloud provider charges it $N \cdot T(h_v + h_a B^a + h_e B^e)$, where h_v is the unit-time VM price, h_a and h_e are unit-time intra- and inter-tenant bandwidth prices. We set h_v, h_a and h_e to 0.02, 0.0001 and 0.0002 such that VM and bandwidth costs have the same order of magnitude. The provider's revenues for three methods are exhibited in Fig. 5(d). Under the target load of 80%, the revenue of DU-OPT is 12.4% and 25.0% higher than that of the DU-GRD and HR-GRD, respectively.

B. Effect of Bandwidth Allocation Strategy

Now we evaluate the effect of *dynamic* bandwidth allocation strategies. The flow-level traffic pattern is configured as follows. First, each VM of a request communicates with a fraction β (0.5 on average) of its rest VMs, forming intra-tenant flows. For example, β is 1 in MapReduce-like requests because of their all-to-all communication. Second, if a request has inter-tenant guarantees, its VMs randomly initialize a batch of inter-tenant flows, the amount of which equals to $20\% - 40\%$ of its total intra-tenant flows [10]. For a tenant request $\langle N, B^a, B^e, S\rangle$ with the expected completion time T, the total intra- and inter-tenant traffic demands are captured by NB^aT and NB^eT. We assume the above demands are evenly distributed over corresponding flows. A request completes when all its flows finish. Here we focus on online requests and run each simulation for 5000 time units. The target load is 80%, which is commonly applied in production datacenters. Currently we adopt an aggressive reallocation time window (1 time unit).

The acceptance ratios for methods with different bandwidth allocation strategies are shown in Fig. 6(a). The dynamic strategy enables providers to accept on average 10% more tenants, because tenants can obtain spare bandwidth to reduce their completion time and release resources to accept more future requests. We notice that the acceptance ratio of each method with the static strategy is slightly less than that with 80% load in Fig. 5(a). The reason stems from the difference in determining when to terminate requests. The actual completion



(a) Accepted requests (b) Completed requests

Fig. 6. Comparison of different bandwidth allocation strategies.

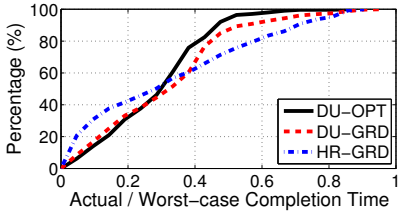


Fig. 7. CDF of the actual / worse-case completion time

time for tenants with traffic demands depends on their traffic patterns and is usually larger than their expected completion time. Using the dynamic strategy, the acceptance ratio of DU-OPT (93.8%) is 5.3% larger than that of HR-GRD (88.5%). That is because under this combination, the average request completion time of DU-OPT is 9.1% less than that of HR-GRD. More requests are thereby completed within the simulation time window, as shown in Fig. 6(b).

Here we also explore the relationship between the actual and worse-case completion time in Fig. 7. For each accepted request, given its traffic pattern and minimum bandwidth guarantees, we can calculate the worse-case completion time for its flows and hence for the entire request. Here we only plot the result for each method with the dynamic strategy. We can find that with DU-OPT, the completion time for 80% tenants is less than 0.4×of the worse-case time. HR-GRD enables a larger fraction of tenants to finish within 0.2×of their worse-case time, whereas it also leads to more tenants with longer completion time (*e.g.*, 40% tenants between 0.4×-1×). This is because the bandwidth allocation strategy HR-GRD uses might result in unbalanced proportions for intra- and inter-tenant flows, as discussed in Sec. II.

VI. Related Work

By employing the hose model that is originally introduced for VPNs [16] , SecondNet [2] provides bandwidth reservations to VM-to-VM flows, while Oktopus [3] provides per-VM bandwidth reservation. [4, 12] extend the hose model to captures specific requirements of cloud applications.

Many work-conserving bandwidth sharing mechanisms are proposed to provide fair sharing of networks for tenants, such as Seawall [5] and Netshare [6]. However, they do not provide deterministic bandwidth guarantees. Gatekeeper [7], EyeQ [8], FairCloud [9] and ElasticSwitch [11] simultaneously achieve minimum bandwidth guarantees and work-conservation.

Ballani et al. [10] explicitly consider bandwidth guarantees for both inter- and inter-tenant communications, and propose a hierarchical hose model. Our work is closely related to the virtual network abstraction in [10]. However, the dual-hose model decouples the guarantees for a tenant's inter-tenant traffic from those for its intra-tenant traffic, which enables cloud providers to accommodate more requests and reduces the completion time of tenants' jobs.

VII. Conclusion

In this paper, we focus on the problem of providing work-conserving bandwidth guarantees in public cloud datacenters. To precisely abstract bandwidth requirements of tenants, we propose the dual-hose model that decouples guarantees for a tenant's inter-tenant traffic from those for its intra-tenant traffic. Based on that, we develop an efficient VM placement algorithm to accommodate tenant requests in physical servers and a bandwidth allocation strategy to achieve work conservation. Through extensive simulation results, we show that our approach benefits both cloud providers and tenants.

References

[1] Amazon Web Services. http://aws.amazon.com/.
[2] 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 *Proc. ACM CoNEXT*, 2010.
[3] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards Predictable Datacenter Networks," in *Proc. ACM SIGCOMM*, 2011.
[4] D. Xie, N. Ding, Y. C. Hu, and R. Kompella, "The Only Constant is Change: Incorporating Time-Varying Network Reservations in Data Centers," in *Proc. ACM SIGCOMM*, 2012.
[5] A. Shieh, S. Kandula, A. Greenberg, and C. Kim, "Sharing the Datacenter Network," in *Proc. USENIX NSDI*, 2011.
[6] T. Lam and G. Varghese, "Netshare: Virtualizing bandwidth within the cloud," *UCSD Technical Report*, 2009.
[7] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes, "Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks," in *Proc. USENIX WIOV*, 2011.
[8] V. Jeyakumar, M. Alizadeh, D. Mazieres, B. Prabhakar, and C. Kim, "EyeQ: Practical network performance isolation for the multi-tenant cloud," in *Proc. USENIX HotCloud*, 2012.
[9] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "FairCloud: Sharing the Network In Cloud Computing," in *Proc. ACM SIGCOMM*, 2012.
[10] H. Ballani, K. Jang, T. Karagiannis, C. Kim, D. Gunawardena, and G. O'Shea, "Chatty Tenants and the Cloud Network Sharing Problem," in *Proc. USENIX NSDI*, 2013.
[11] L. Popa, P. Yalagandula, S. Banerjee, J. Mogul, Y. Turner, and R. Santos, "ElasticSwitch: Practical Work-Conserving Bandwidth Guarantees for Cloud Computing," in *Proc. ACM SIGCOMM*, 2013.
[12] J. Lee, Y. Turner, M. Lee, L. Popa, J. Kang, S. Banerjee, and P. Sharma, "Application-Driven Bandwidth Guarantees in Datacenters," in *Proc. ACM SIGCOMM*, 2014.
[13] J. Schad, J. Dittrich, and J.A. Quiane-Ruiz, "Runtime measurements in the cloud: observing, analyzing, and reducing variance," *VLDB*, 2010.
[14] P. Bodik, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica, "Surviving failures in bandwidth-constrained datacenters," in *Proc. ACM SIGCOMM*, 2012.
[15] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. ACM IMC*, 2010.
[16] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. Merive, "A flexible model for resource management in virtual private networks," in *Proc. ACM SIGCOMM*, 1999.
[17] K. Kwong, R. Guerin, A. Shaikh, and S. Tao, "Balancing performance, robustness and flexibility in routing systems," *ACM CoNEXT*, 2008.
[18] Supplementary File. http://www.thucsnet.org/uploads/2/5/2/8/25289795/main-supp.pdf.