FairCloud: Sharing The Network In Cloud Computing

Lucian Popa[*†] Arvind Krishnamurthy[‡] Sylvia Ratnasamy[*] Ion Stoica[*]

ABSTRACT

The network is a crucial resource in cloud computing, but in contrast to other resources such as CPU or memory, the network is currently shared in a best effort manner. However, sharing the network in a datacenter is more challenging than sharing the other resources. The key difficulty is that the network allocation for a VM X depends not only on the VMs running on the same machine with X, but also on the other VMs that X communicates with, as well as on the cross-traffic on each link used by X. In this paper, we first propose a set of desirable properties for allocating the network bandwidth in a datacenter at the VM granularity, and show that there exists a fundamental tradeoff between the ability to share congested links in proportion to payment and the ability to provide minimal bandwidth guarantees to VMs. Second, we show that the existing allocation models violate one or more of these properties, and propose a mechanism that can select different points in the aforementioned tradeoff between payment proportionality and bandwidth guarantees.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General

General Terms

Design, Algorithms

1. INTRODUCTION

Cloud computing represents the platform of choice for deploying and running many of today's businesses. Core to cloud computing is the ability to *share and multiplex* resources across users. One critical resource is the network, as an application's performance can be significantly impacted by the share of network capacity it receives. Today, however, cloud networks are shared in a best-effort manner making it hard for both users and cloud operators to reason about how network resources are allocated.

[*]UC Berkeley, {popa,sylvia,istoica}@cs.berkeley.edu
[†]HP Labs
[‡]Univ. of Washington, arvind@cs.washington.edu

We argue that a more desirable approach to sharing the network would meet two key objectives. The first is that network resources be shared in proportion to payment. Today, payment is at the granularity of a virtual machine (VM) and hence if (for example) VMs belonging to two users X and Y compete for a congested link L, then we would want to see that L's bandwidth is allocated between X and Y in the ratio $\frac{P_X}{P_Y}$ where P_X and P_Y denote the payment by users X and Y respectively. A second desirable objective is that users receive some guarantees on the minimal network bandwidth they can expect when buying VMs irrespective of the network utilization of other users. This objective is commonly achieved today for other cloud resources such as CPUs, memory and disk and is key to achieving predictability in application performance. As we shall show, traditional mechanisms for sharing the network – *e.g.,* fairness between flows, source-destination pairs, or sources alone – cannot meet either of the above desirable objectives.

In this paper, we address the question of how the network in a cloud environment should be shared to achieve the above objectives. We start with developing a model for network sharing that rigorously captures the above objectives, and then propose several mechanisms to implement the model. Defining such a model is challenging as the network differs from other resources in two key aspects:

1. Interdependent Users: With resources such as CPU or memory one can clearly attribute the consumption of a unit of resource to a single VM. In contrast, network communication involves at least two VMs—a source and a destination. Which of the two VMs should consumption be attributed to? At one extreme, one can attribute consumption entirely to the source and consequently make bandwidth allocation decisions with respect to sources; at the other extreme one might do the same with respect to only destinations. Prior models for resource sharing have often *implicitly* assumed a particular design point; *e.g.,* RSVP [19] is destination driven, while Diffserv [13] and Seawall [17] adopt a source based approach. We argue that the appropriate approach for cloud networks is to consider *both* sources and destinations in making allocation decisions. A further complication is that a source (destination) VM may simultaneously communicate with multiple destination (source) VMs and hence one has to consider the boarder communication pattern of a VM. In short, the network allocation for a VM depends on the *other* VMs it communicates with.

2. Interdependent Resources: For resources such as CPU and memory, it is relatively easy to take the total aggre-

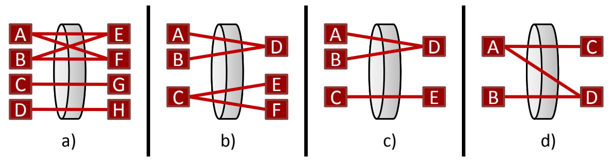

Figure 1: Sharing a single link, examples. A square represents a VM while a line connecting them represents a communication. Each VM can belong to any tenant.

gate resource and divide it into smaller units (cores, cycles, pages) that can be *independently* allocated to different VMs. The network however is formed by a topology of interconnected links and thus is not easily divisible into independent units for allocation. In short, the network allocation for a VM A depends on *any* VM whose traffic shares a link with traffic to/from A.

Building on the above observations, our contributions in this paper are as follows. First, we propose a set of desirable properties for allocating the network bandwidth in a datacenter at the VM granularity, and show that there exists a fundamental tradeoff between the ability to share congested links in proportion to payment and the ability to provide minimal bandwidth guarantees to VMs. Second, we show that the existing allocation models violate one or more of these properties, and propose a mechanism that can select different points in the aforementioned tradeoff between payment proportionality and bandwidth guarantees.

Before proceeding, we clarify some of the assumptions our work builds on. We assume an infrastructure as a service (IaaS) cloud model where users pay per VM [1] and hence our goals for network sharing are defined from a per-VM viewpoint, akin to how other cloud resources are allocated today (we compare this to alternate approaches in §5). Our discussion is *agnostic* to VM placement and routing algorithms which we assume are implemented independently, and we only consider network sharing in a single datacenter. Finally, our discussion is largely orthogonal to work on network topologies to improve bisection bandwidth [2, 10, 12], as the possibility of congestion (and hence the need for sharing policies) remains even in full bisection bandwidth networks—*e.g.* many-to-many communication, as in map-reduce, can congest the links of source or destination VMs.

2. CHALLENGES SHARING NETWORKS

In this section, we discuss the challenges associated with sharing cloud networks in the context of previous proposals.

The traditional approach to sharing network resources is to perform *per flow* fairness, where a flow is characterized by the standard five-tuple in packet headers. However, per flow allocation could lead to unfair bandwidth allocation at the VM (endpoint) granularity [6]. Indeed, two VMs can increase the traffic allocation between them at the expense of other VMs by simply instantiating more flows.

A natural "fix" for the per-flow allocation unfairness would be to use a *per source-destination pair* (per S-D pair) allocation model, where each source-destination pair is allocated an equal share of a link's bandwidth regardless of the number of flows between the pair of VMs. However, this model is still arguably unfair, as a VM that communicates with many VMs gets more bandwidth than a VM that communicates with fewer VMs. For example, a user that employs an all-to-all communication pattern between N VMs will get a bandwidth share of $O(N^2)$, while a user that performs one-to-one communication between the same number of N VMs will get a share of only $O(N)$. Fig. 1 (a) shows one such example, where the allocation of hosts `A,B,E,F` is twice that of hosts `C,D,G,H`.

To address this problem, previous works have proposed using a *per source* allocation model, *e.g.*, Seawall [17]. With such a model, sources communicating over a given link are assigned equal weights, and the traffic is divided fairly between sources. While this model is fair to sources, it might not be fair to destinations. For example, in Fig. 1 (b) `E` and `F` receive four times less traffic than `D`, and for traffic flowing in the opposite direction, `A` and `B` receive four times less traffic than `C`.

Similarly, while a *per destination* allocation model has some desirable properties, such as providing protection against DoS attacks [18], it is not fair to sources. Thus, these allocation models are *asymmetric* in that they can only be fair with either sources or destinations, but not both. Such allocation asymmetry is undesirable, as it implicitly assumes that the network has knowledge about whether a VM values more the incoming or the outgoing traffic, which is not the case in practice. All examples in Fig. 1 (b), (c) and (d) suffer from asymmetry, as the allocation for each host and each S-D pair will be different along the two directions of the link.

From the above examples, we conclude that the "obvious" approaches to achieving fairness do not offer satisfactory solutions for sharing network resources in the cloud. Further, these examples illustrate that defining what is the "fair" share of the network is not a trivial question even for simple scenarios. Thus, to answer this question, we start with formulating a set of potentially desirable properties regarding network bandwidth allocation before discussing ways to achieve them.

3. PROPERTIES FOR NETWORK SHARING

Table 1 presents five basic properties that we argue should be implemented by any mechanism for sharing a datacenter network. In addition to defining these properties, the table also provides brief justifications for why these properties are desirable. The schemes outlined in the previous section violate some of these properties. For instance, per flow based allocation is not strategy proof, while per source and per destination allocations do not provide the symmetry property.[1]

In addition to the basic properties outlined above, we also

[1]Similar properties to our first two properties were also identified as desirable when sharing multiple different resource types [8].

Property	Definition	Motivation
B1. Strategy Proofness	A set of VMs Q should not be able to increase its bandwidth allocation to another set of VMs P by modifying its behavior at the application level (*e.g.*, using multiple flows or adopting a different traffic pattern).	This property prevents VMs from obtaining an unfair bandwidth allocation with respect to competing VMs.
B2. Pareto Efficiency	If the traffic between VMs X and Y is bottlenecked at link L, then it should not be possible to increase the allocation for $X - Y$ without decreasing the allocation to another source-destination pair using the same link.	If this property is not satisfied, the network is not fully utilized even when there is unsatisfied demand.
B3. Non-zero Flow Allocation	Each pair of VMs desiring to communicate should obtain a non-zero bandwidth allocation irrespective of the overall communication pattern in the network.	Users expect a strictly positive bandwidth allocation between every pair of VMs even if they are generating other flows.
B4. Independence	The bandwidth allocations for a VM along two paths that share no congested links should be independent. In particular, if a VM sends traffic on an uncongested path, this should not affect its traffic on other congested paths.	This is a property that is satisfied in today's Internet. Lack of this property would lead to inefficient utilization; for example, an endpoint might refrain from sending on an uncongested path in order to get a larger traffic share on a different congested path.
B5. Symmetry	Assume all links in the network have the same capacity in both directions. If we switch the directions of all flows in the network, then the reverse allocation of each flow should match the original (forward) allocation of that flow.	Existing allocation models make an implicit assumption as to whether the allocation is receiver or sender centric; however, in general, it is difficult to anticipate application-level preferences. For example, server applications might value outgoing traffic while client applications might value incoming traffic. In the absence of application-specific information, we prefer allocations that provide equal weight to both incoming and outgoing traffic.

Table 1: Basic sharing properties desirable for any bandwidth allocation mechanism in clouds.

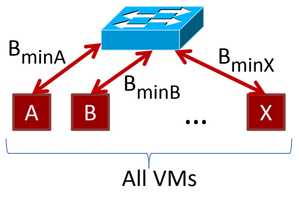

Figure 2: Model for Guaranteed Bandwidth.

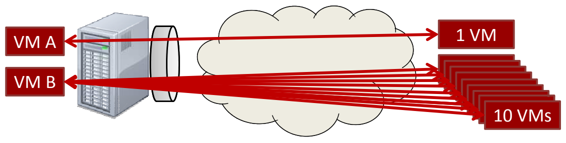

Figure 3: Sharing the access link of one machine.

examine additional properties required to support VMs with heterogeneous configurations. Just as today's cloud providers offer VMs with different CPU and memory configurations at different prices, we consider a setting where each VM has a (positive) *network weight* associated with it based on the user's payment.

Table 2 lists the two properties that we propose for this usage model. The first property describes three increasing degrees of correlating bandwidth allocation with weights: monotonicity, strict monotonicity, and proportionality. Proportional allocation provides highest sensitivity to weights and is illustrated by the following example. Consider Fig. 1 (c) assuming equal weights per VM. In this case, if there is sufficient demand for all flows, A-D and B-D flows would in total receive $3/5$ of the link capacity, while C-E would receive $2/5$, and this allocation is proportional to the number of VMs in each communication set.

The second property, guaranteed bandwidth, enables predictability for the applications deployed in clouds. For example, if a user rents two VMs (A and B), the user is provided a lower-bound on the bandwidth that will be allocated for the traffic between the two VMs, irrespective of the communication demands of the other VMs in the network. The guaranteed bandwidth in this case is $\min(B_{minA}, B_{minB})$. As shown in Fig. 2, this is equivalent to the network resource sharing provided by a star-topology network where each VM is connected to a central switch using a link with capacity equal to that of the VM's minimum guaranteed bandwidth.

The core observation of this section is that the two properties listed in Table 2 (weight fidelity and guaranteed bandwidth) are *conflicting* and illustrate a key tradeoff in sharing the network. Specifically, the higher the fidelity of respecting weights, the smaller are the bandwidth guarantees.

The example in Fig. 3 helps illustrate this tradeoff. A and B are two VMs collocated on the same server, and A communicates with one other VM while B communicates with 10 others. Let all VMs have unit weights. If we consider a proportional allocation, A gets only $2/13$ of the access link (as there are 13 VMs in total competing for the access link). It is easy to see that if B communicates with more VMs, A's share of its access link would further decrease. A strictly monotonic allocation might provide a larger share of the access link capacity to A compared to a proportional allocation, but it would also reduce A's share when any of the remote hosts communicating with B increase their weight. However, in a different model that satisfies only monotonicity, A could be guaranteed a sizable fraction of its access link capacity, regardless of the communication pattern of B. For example, in a network with full bisection bandwidth, A could be guaranteed half of the access link capacity to communicate with other VMs in the network. In this case, the flows of A and B would each get $\frac{1}{2}$ of the access link, which is not achievable while having strict monotonicity, let alone proportionality.

Property	Definition	Motivation
W1. Weight Fidelity - increasing degrees of respecting weights		
a. Monotonicity	If the weight associated with a VM is increased, all its traffic allocations through the network either increase or remain unchanged, *i.e.,* the allocations do not decrease.	If the allocation mechanism does not exhibit this property, then the system does not provide sufficient incentives to customers to rent VM instances with higher weights.
b. Strict Monotonicity	If the weight associated with a VM is increased, all its traffic allocations through the network increase, assuming that the VM has unsatisfied demand.	This provides stronger incentives than *monotonicity*, particularly when we have inter-tenant traffic. For example, consider a tenant A that is using the service provided by another tenant B, whose network flows are bottlenecked. With this property, tenant A would always get a higher bandwidth by buying higher weight VMs, at the expense of other tenants sharing the access links near B.
c. Proportionality	On any congested link L actively used by a set T of VMs, any subset $Q \subset T$ that communicates only to other VMs in Q (*i.e.,*Q does not communicate with $T \setminus Q$) is allocated at least a total share of W_Q/W_T of the bandwidth, where W_Q is the total weight of the VMs in set Q, assuming all VMs in Q have unsatisfied demand. This allocation should occur regardless of the distribution of the VMs in the set Q between the two ends of the link and of the communication pattern over L.	This property can be seen as providing network shares that are proportional to payment. For example, if all VMs have equal weight and we have one tenant with k_1 VMs and another tenant with k_2 VMs that compete over L, then the ratio of the bandwidths allocated to them is k_1/k_2.
W2. Guaranteed Bandwidth	Each VM X is guaranteed a bandwidth allocation of B_{minX}, as if X were connected by a link of capacity B_{minX} to a central switch with infinite capacity to which all other VMs are also connected (see Fig. 2). This is also known as the hose model [7].	There is a lower bound on the bandwidth allocated to X regardless of the traffic demands and the communication patterns of the other VMs. This property enables predictability in tenant applications. For example, if one knows the communication pattern between her VMs, she can select the weights accordingly and predict the application performance. Higher guaranteed bandwidths provide stronger incentives for tenants to rent VMs with higher weights.

Table 2: Desirable properties for a per VM payment model using weights.

Thus, one can achieve higher bandwidth guarantees but be less sensitive to weight allocation, or respect weights strictly on each link but provide very small bandwidth guarantees.

Strictly speaking, the guaranteed bandwidth property is achieved by any allocation, since the number of contenders is always bounded in practice (*e.g.,* even with a per flow allocation, there is a practical limit to the number of flows used). What we would like is that the minimal guarantee be a useful metric, since, for example, dividing a single link's capacity equally across all VMs could lead to a meaningless guarantee. Ideally, we would like the minimum bandwidth guarantee to be comparable to the bisection bandwidth of the network divided by the number of VMs in the network, *i.e.,* if we increase the network size and scale the bisection bandwidth by a similar factor, then the guaranteed bandwidth would remain the same. We also note that preserving the value of the guaranteed bandwidth during one VM's lifetime relies on a form of admission control of VMs and weights into the network (the number of VMs per server is anyway limited by the number of CPUs, memory, *etc.*).

The first three columns of Table 3 illustrate the properties provided by the traditional sharing mechanisms. The per flow allocation model does not respect any of the weight fidelity properties since its allocation can change simply based on the number of application level flows. The per source allocation model provides non-strict monotonicity, but does not provide adequate bandwidth guarantees for the incoming traffic. In the next section, we discuss how one can achieve all of the non-conflicting properties and at the same time explore different tradeoff points between weight fidelity and guaranteed bandwidth.

4. MECHANISM

We now present *Per Endpoint Sharing* (*PES*), a flexible mechanism for sharing cloud networks that allows one to explicitly trade between weight fidelity and guaranteed bandwidth.

To satisfy the non-zero flow allocation property, a natural approach is to assign each source-destination pair a weight that depends on the weights of the source and the destination, *i.e.,* $W_{S-D} = f(W_S, W_D)$, where W_S is the weight of endpoint S. To satisfy the symmetry property, the weight of the allocation should be the same in both directions, *i.e.,* $W_{A-B} = W_{B-A}$.

***Per Endpoint Sharing* (*PES*)** is a mechanism that assigns to a communication between VMs A and B on link L a weight of:

$$W_{A-B} = \frac{W_A}{N_A} + \frac{W_B}{N_B}$$

where N_A is the number of other VMs A is communicating with on link L (similarly N_B). The weights can be normalized by dividing them with W_T, the total weight of all VMs communicating on that link.

The *PES* algorithm achieves weight *proportionality* as described in Table 2. For example, consider Fig. 1 (c) and assume that all VMs have unit weight. *PES* would assign weights of 1.5, 1.5, and 2 to A-D, B-D, and C-E, respectively. So the flows between A-D and B-D would receive $\frac{1.5}{5}$ of the link capacity, since D's weight is split across its two flows. In Fig. 1 (d), *PES* would assign weights of 1.5, 1.5, and 1 to A-C, B-D, and A-D, respectively. The flow between A and D would then receive $\frac{1}{4}$ of the link capacity.

Note that a drawback of the *PES* mechanism, common to all mechanisms that compute a static weight for each source

Property \ Mechanism	**Per Flow**	**Per S-D Pair**	**Per Source (Per Dest)**	***PES***	***OSPES***
B1. Strategy Proofness	$\times$	$\times$	$\surd$	$\surd$	$\surd$
B2. Pareto Efficiency	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
B3. Non-zero Flow Alloc.	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
B4. Independence	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
B5. Symmetry	$\surd$	$\surd$	$\times$	$\surd$	$\surd$
W1. Weight Fidelity	$\times$	Monotonicity	Monotonicity	Proportionality	Monotonicity
W2. Guaranteed Bandwidth	$\times$ (none)	$\times$ (very small, $\approx BB/{N_T}^2$)	$\times$ (very small, $\approx C/N_T$)	$\times$ (very small, $\approx C \cdot W/W_T$)	$\surd$ (max, $\approx \min(C \cdot W/W_L, BB \cdot W/W_T)$)

Table 3: Properties achieved by different network sharing mechanisms. The guarantees are discussed in the context of a tree-based topology. Notation: C = access link capacity, N_T = total number of VMs in the network, W = weight of VM in discussion, W_T = weight of all VMs in the network, W_L = weight of all VMs collocated with VM in discussion, BB = bisection bandwidth.

destination pair, is that the properties exhibited by the allocation are different for different demands. For example, consider Fig. 1 (c) and assume that all VMs have unit weight. If the flow between A and D has a very small demand ϵ the allocation between B-D and C-E will respect the ratio of $1.5/2$ instead of a more desirable $1/1$ ratio. Clearly, this could be addressed if we also take into account the actual demands, however, such a mechanism will in practice be more difficult to apply, and we leave its exploration to future work.

Table 3 illustrates how *PES* satisfies the properties discussed in the previous section. Note that the theoretical bandwidth guarantee offered by *PES* is small since, in the worst case, one VM might have to divide its access link fairly with all the other VMs in the network. However, if the congestion is in the center of the network (as it typically occurs) and the routing can balance the traffic across all available paths, a VM would in fact get its fair share of the bisection bandwidth.

To offer higher worst case bandwidth guarantees, intuitively we would like to give higher importance to some VMs compared to others based on the "importance" of the link with respect to the VMs. For example, on the access link of one host, we would like to divide the link in a proportion closer to the VMs on that host rather than to the remote VMs.

To this end, we generalize *PES* to $W_{A-B} = W_{B-A} = \alpha\frac{W_A}{N_A} + \beta\frac{W_B}{N_B}$. The coefficients α and β provide the ability to weight differently the VMs located on the two sides of the link: α is applied to all the VMs on one side of L while β to the VMs on the other side. In this way, the weights of the VMs on one side of L can be scaled up/down or even completely disregarded by using different values for α and β.

By setting specific values for α and β at different links in the network, one can use the generalized *PES* mechanism to achieve different design points along the the described tradeoff, *e.g.*, provide higher bandwidth guarantees but with low weight fidelity or provide lower bandwidth guarantees but with high weight fidelity. Due to space constraints, we now discuss only one simple design point suitable for tree-based topologies (*e.g.*, traditional datacenter architectures, VL2 [9], fat-trees [2]).

***One Sided PES* (*OSPES*)** is a mechanism that prioritizes VMs that are close to a given link. More precisely, *OSPES* uses $\alpha = 1$ and $\beta = 0$ for all links in the tree that are closer to A than B and $\alpha = 0$ and $\beta = 1$ for all links closer to B than A. Essentially, *OSPES* translates into applying per source fair sharing for the traffic towards the tree root and per destination fair sharing for the traffic from the root. Table 3 summarizes the properties achieved by *OSPES*. In a full bisection bandwidth network, each VM is guaranteed a bandwidth that represents its fair share of the access link when competing *only* with the other VMs collocated at the same host. We assume here that the total weight of the VMs collocated on each host is the same throughout the network. For example, in Fig. 3, if A and B have equal weights, A would be allocated at least $\frac{1}{2}$ of its access link capacity regardless of how many other VMs communicate with B. If the access link is 1 Gbs, then each VM is guaranteed 500 Mbps. This result can be generalized to tree network topologies with an oversubscription factor of k, *e.g.*, for a network with an oversubscription ratio of 2:1, each VM is guaranteed 250 Mbps. Here, we assume that if two VMs can communicate via multiple paths, the routing protocol can load balance the VMs' traffic across all the available paths. This assumption holds for many of the newly proposed multi-tree topologies that use multi-path routing to fully utilize the network bisection bandwidth, *e.g.*, [2, 3, 9, 15].

OSPES only achieves non-strict monotonicity, since increasing the weights on the less important side of the link does not increase the allocation. By setting specific values for α and β, one can achieve strict monotonicity and half of the bandwidth guarantee offered by *OSPES*. Moreover, this approach can be generalized to other types of networks (such as BCube [10], DCell [12], *etc.*), but due to space constraints we omit these results.[2]

Lastly, we note that one might consider as alternative to *PES* to apply max-min [5] at the granularity of VMs (max-min is the most common mechanism for allocating bandwidth between flows). Max-min fairness could be applied at

[2]At a high level, the idea is to equalize the importance of the VMs situated on the two ends of each link. Thus, each VM has at least a guaranteed bandwidth of $\frac{1}{2}$ its share of the bisection capacity, while the allocation is strictly monotonic at the same time.

the granularity of VMs by maximizing the minimum allocation for the traffic sent or received by any VM. For example, assuming equal weight VMs and infinite traffic demands in Fig. 1 (b), VMs A, B, E and F would each get an equal allocation to send/receive $1/4$ of the link capacity. While this approach is fair at the VM level and it satisfies the symmetry property, it may lead to some flows not getting any traffic at all, which would violate the non-zero flow allocation property. For example, in Fig. 1 (d), per VM max-min fairness would allocate one half of the capacity to the flows between A and C and one half to the flows between B and D. This is a perfect distribution of the capacity between the VMs (each would send/receive an equal amount), but there is no traffic between A and D. This allocation is also not strictly monotonic and does not provide adequate bandwidth guarantees.

Practical Challenges: The *PES* mechanisms can be implemented at switches in the network. It could also be implemented using a centralized controller and enforced at hypervisors instead of switches. However, we leave the details of implementing *PES* to future work. To learn the weight of a VM at a switch, providers can implicitly encode weights into different sets of IP or (virtual) MAC addresses or could use the QoS bits in the packet header filled out by hypervisors (for security reasons). An alternative is to use a centralized controller to inform switches when a new VM is instantiated.

5. RELATED WORK

Recently, there have been a few proposals on how to share the network within a datacenter. Seawall [17] proposes to enforce fairness in hypervisors based on ECN feedback from switches, however it uses per source sharing. Gatekeeper [16] proposes a per-VM hose model similar to our guaranteed bandwidth model for full bisection-bandwidth networks. Gatekeeper uses a hypervisor-based approach and provides guarantees for sharing access links. We are currently investigating whether some variants of our more general sharing mechanism can be implemented in hypervisors as well.

Other related work propose bandwidth allocations at the granularity of tenants rather than VMs. We divide these approaches into two broad categories: (a) reserving virtual networks for each tenant [4, 11] and (b) network multiplexing along with per tenant weights [14]. We discuss these schemes below.

Reserving virtual networks per tenant as proposed by Oktopus [4] and SecondNet [11] does provide bandwidth guarantees, but only when communicating with other VMs of the same tenant. More importantly, a reservation system does not achieve the Pareto Efficiency property, since the unused bandwidth is not shared between tenants. For small tenants in oversubscribed networks, per tenant reservation can offer higher bandwidth guarantees than per VM sharing, by reserving clusters of nodes that are collocated on the same or nearby racks. However, per VM sharing discussed in this paper could also be extended to offer different bandwidth guarantees to different sets of VMs (*e.g.,*VMs rented by the same tenant vs. other VMs), when it is integrated with VM placement. And while specifying a virtual topology could provide fine grain control to users, the allocation at a VM level is simpler. Users do not need to also specify network topologies and to possibly update them when adding new VMs (especially since adding machines on a daily/hourly basis is common).

Schemes that advocate network multiplexing through the use of per tenant weights (*e.g.,* NetShare [14]) provide a different set of properties than those that use per VM weights. The number of tenant VMs that communicate over a congested link varies across links and is not necessarily reflected by the per tenant weight, which is a network wide constant. Thus, the properties achieved by a placement agnostic network sharing mechanism are more difficult to understand.

6. CONCLUSION

In this paper we address the problem of sharing the network within a cloud computing datacenter. We propose to extend today's *per VM* model of sharing cloud resources, such as CPU and memory, also to network resources. To this end, we propose a set of properties to be implemented by the network sharing mechanism. We also identify a key tradeoff between the ability to share congested links in proportion to payment and the ability to provide minimal bandwidth guarantees to communicating VMs. Finally, we present a mechanism that is able to achieve different design points in the tradeoff between these conflicting requirements.

7. REFERENCES

[1] Amazon web services. `http://aws.amazon.com`.
[2] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *SIGCOMM*. ACM, 2008.
[3] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic Flow Scheduling for Data Center Networks. In *NSDI*, 2010.
[4] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards Predictable Datacenter Networks. In *ACM SIGCOMM*, 2011.
[5] D. P. Bertsekas and R. Gallager. *Data networks (2. ed.).* Prentice Hall, 1992.
[6] B. Briscoe. Flow rate fairness: Dismantling a religion. *ACM SIGCOMM Computer Communication Review*, 2007.
[7] N. G. Duffield, P. Goyal, A. G. Greenberg, P. P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe. A flexible model for resource management in virtual private networks. In *SIGCOMM*, 1999.
[8] A. Ghodsi, M. Zaharia, B. Hindman, et al. Dominant resource fairness: fair allocation of multiple resource types. In *USENIX NSDI*, 2011.
[9] 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. *ACM SIGCOMM*, August 17 - 21 2009.
[10] C. Guo and G. Lu *et al.* BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers. *ACM SIGCOMM*, 2009.
[11] C. Guo and G. Lu *et al.* Secondnet: a data center network virtualization architecture with bandwidth guarantees. In *CoNEXT*. ACM, 2010.
[12] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. Dcell: A Scalable and Fault-tolerant Network Structure for Data Centers. In *SIGCOMM*, 2008.
[13] Internet Draft. RFC2475 - An Architecture for Differentiated Services, 1998.
[14] T. Lam, S. Radhakrishnan, A. Vahdat, and G. Varghese. NetShare: Virtualizing Data Center Networks across Services. *Technical Report, UCSD*, 2010.
[15] C. Raiciu and S. Barre *et al.* Improving Datacenter Performance and Robustness with Multipath TCP. In *ACM SIGCOMM*, 2011.
[16] H. Rodrigues, J. R. Santos, Y. Turner, et al. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. In *USENIX WIOV*, 2011.
[17] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha. Sharing the Data Center Network. In *Usenix NSDI*, 2011.
[18] X. Yang, D. J. Wetherall, and T. Anderson. A DoS-limiting Network Architecture. In *ACM SIGCOMM*, 2005.
[19] L. Zhang, S. E. Deering, D. Estrin, S. Shenker, and D. Zappala. Rsvp: A new resource reservation protocol. *IEEE Network*, 1993.