FairCloud: Sharing the Network in Cloud Computing

Lucian Popa
HP Labs

Gautam Kumar
UC Berkeley

Mosharaf Chowdhury
UC Berkeley

Arvind Krishnamurthy
U. Washington

Sylvia Ratnasamy
UC Berkeley

Ion Stoica
UC Berkeley

ABSTRACT

The network, similar to CPU and memory, is a critical and shared resource in the cloud. However, unlike other resources, it is neither shared proportionally to payment, nor do cloud providers offer minimum guarantees on network bandwidth. The reason networks are more difficult to share is because the network allocation of a virtual machine (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 and the cross-traffic on each link used by *X*. In this paper, we start from the above requirements–payment proportionality and minimum guarantees–and show that the network-specific challenges lead to fundamental tradeoffs when sharing cloud networks. We then propose a set of properties to explicitly express these tradeoffs. Finally, we present three allocation policies that allow us to navigate the tradeoff space. We evaluate their characteristics through simulation and testbed experiments to show that they can provide minimum guarantees and achieve better proportionality than existing solutions.

Categories and Subject Descriptors

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

Keywords: cloud computing, network sharing

1. INTRODUCTION

Cloud computing is the platform of choice for deploying and running many of today's businesses. Central to cloud computing is its ability to share and multiplex resources across multiple tenants. Cloud networks, however, are shared in a best-effort manner making it hard for both tenants and cloud providers to reason about how network resources are allocated.

We argue that a desirable solution for sharing cloud networks should meet three requirements. The first is to provide tenants guarantees on the minimum network bandwidth they can expect for each VM they buy, irrespective of the network utilization of other tenants. Such guarantees are common for resources like CPU and memory, and having the same for the network is key to achieving lower bounds for the worst-case performance of an application. We refer to this requirement as *min-guarantee*.

The second desirable requirement, referred to as *high utilization*, aims to maximize network utilization in the presence of unsatisfied demands. For example, we would like an application to use the entire available bandwidth when no other application is active. This can significantly improve the performance for applications with bursty (on/off) traffic patterns, such as MapReduce. Improving application performance may, in turn, allow the provider to charge more.

The third and last requirement is that network resources should be divided among tenants in proportion to their payments, similar to CPU or memory. Under the current flat-rate per VM payment model, this means that two tenants with the *same* number of (identical) VMs should get the *same* aggregate bandwidth assuming they both have sufficient demands, since they paid the *same* amount of money. We refer to this allocation requirement as *network proportionality*. Note that the min-guarantee requirement does not achieve network proportionality, as it only refers to the *minimum* bandwidth guarantee of VMs; however, a VM can get a lower allocation than its guarantee when it has a lower demand, or a higher allocation when the other VMs have lower demands than their guarantees.

Unfortunately, none of the traditional network sharing policies (e.g., fairness among flows, source-destination pairs, or sources alone) can meet either of the min-guarantee or network proportionality requirements, while more recent proposals such as Oktopus [10] can only provide minimum guarantees. We argue that the difficulty of developing solutions to achieve these requirements stems from the following *fundamental tradeoffs*:

- There is a hard tradeoff between min-guarantee and network proportionality: if one aims to achieve min-guarantee, she cannot achieve network proportionality, and vice versa.
- Even without requiring min-guarantees, there is a tradeoff between network proportionality and high utilization.

To this end, we propose a set of properties to help us navigate the tradeoff space and present three allocation policies that obtain the maximal sets of non-conflicting desirable properties. There are two key concepts at the core of these policies. First, we allocate bandwidth along congested links in proportion to the number of VMs of each tenant, not to the number of flows, sources, or source-destination pairs of the tenant. This allows us to meet (restricted versions of) the network proportionality requirement. Second, we use the VM proximity to a link to compute a tenant's share on that link. Specifically, in tree-based network topologies, the share of a tenant on a link is computed as the number of VMs of that tenant in the sub-tree delimited by that link. This allows us to provide minimum guarantees by trading off network proportionality.

In summary, we make two contributions in this paper.

1. We expose the fundamental tradeoffs in network resource allocation in cloud and data center environments, and we provide a set of requirements and properties that allow us to explicitly express these tradeoffs (§2 and §3).
2. We develop a set of resource allocation policies to best navigate these tradeoffs (§4) and evaluate them using simulation and testbed experiments (§5).

$\mathbf{D_A}$	Demand matrix of tenant A
$\mathbf{R_A}$	Allocation matrix for tenant A
$\lvert\mathbf{R_A}\rvert$	Aggregate bandwidth of tenant A

Table 1: Notation

2. REQUIREMENTS AND TRADEOFFS

In this section, we elaborate on the desirable requirements for bandwidth allocation across multiple tenants. Then, we show that, even though all these requirements are desirable at the same time, they cannot be simultaneously achieved. In this context, we discuss tradeoffs between these requirements and place traditional allocation policies in this space.

2.1 Assumptions and Notation

We assume an Infrastructure-as-a-Service (IaaS) cloud model such as Amazon EC2 [1], where tenants pay a fixed flat-rate per VM. Hence, our goals for network sharing are defined from a per VM viewpoint, akin to how other cloud resources are allocated today. For the simplicity of exposition, we assume in this section that all VMs are identical (in terms of hardware resources) and have the same price. We will consider heterogeneous VMs and expand on alternate pricing models in §4.

Our discussion is agnostic to VM placement and routing, which we assume are implemented independently. Also, our work is largely orthogonal to work on network topologies aimed at improving bisection bandwidth [7, 16, 18], because the possibility of congestion (and hence the need for sharing policies) remains even in full bisection bandwidth networks (e.g., many-to-many communication in MapReduce can congest any of the links in the network).

We abstract a cloud provider's network by a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is the set of physical machines and $\mathbf{E}$ is the set of links that connect them. A machine is either a switch/router or a server. A server can host one or more VMs, possibly belonging to different tenants. We use the term congested to refer to a fully utilized link.

A tenant, K, consists of N_K VMs, and has an instantaneous $N_K \times N_K$ demand matrix $\mathbf{D_K} = [D_K^{i,j}]$, where $D_K^{i,j}$ represents the bandwidth demand from tenant K's VM i to VM j. An allocation policy P allocates a set of rates $\mathbf{R} = \{\mathbf{R}_1, \ldots, \mathbf{R_m}\}$ to the set of m tenants with demands $\mathbf{D} = \{\mathbf{D}_1, \ldots, \mathbf{D_m}\}$, i.e.,

$$P(\mathbf{G}, \mathbf{D}) = \{\mathbf{R}_1, \ldots \mathbf{R_m}\}, \quad (1)$$

where the $N_K \times N_K$ matrix $\mathbf{R_K} = [R_K^{i,j}]$ is the instantaneous bandwidth allocation for tenant K and $R_K^{i,j} \leq D_K^{i,j}\ \forall i, j$. Finally, let $|\mathbf{R_K}| = \sum_{i,j} R_K^{i,j}$ denote the aggregate bandwidth allocation of tenant K (see Table 1).

2.2 Allocation Requirements

We desire a bandwidth allocation policy that meets the following three requirements.

• ***Min-Guarantee*:** Provide a minimum absolute bandwidth guarantee for each VM. We consider the hose model [13] shown in Figure 1, where each VM is connected to a non-blocking switch by a dedicated connection whose capacity is equal to the minimum bandwidth guarantee. A similar model is assumed by other efforts such as Oktopus [10] or Gatekeeper [24]. This requirement is key for achieving predictable application performance and is usually enforced through admission control.

• ***High Utilization*:** Do not leave network resources underutilized when there is unsatisfied demand. Consider a statically reserved, non work-conserving hose model in which tenants cannot exceed their guaranteed bandwidth; while it is a good fit for low-latency

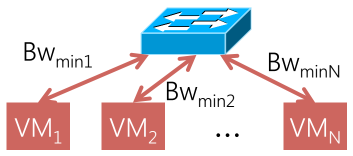


Figure 1: Guaranteed bandwidth of each VM in the hose model.

and predictable traffic, it is not well suited for bursty traffic. Note that high utilization is more general than work conservation; for instance, as we will discuss in §2.4, tenants could be disincentivized to use free network resources even if they have unsatisfied demands. This would lead to a lower utilization of the network even for a work conserving allocation.

High utilization is particularly important for throughput-sensitive applications. For example, a tenant running MapReduce jobs can utilize excess bandwidth (i.e., bandwidth unused by other tenants) to improve the completion times for her jobs.[1]

• ***Network Proportionality*:** Share bandwidth between tenants based on their payments, just as any other resource in the cloud. Given today's flat-rate per VM payment and assuming the VMs to be identical, this requires the network share of a tenant to be proportional to the total number of her VMs. Thus, given two tenants A and B, an ideal solution for network proportionality would allocate $|\mathbf{R_A}|/|\mathbf{R_B}| = N_A/N_B$. Unfortunately, this is not always possible due to different communication patterns, capacity constraints and demands of the tenants. For example, assume $N_A = N_B$, and both tenants have infinite demands, but all tenant A's traffic traverses a link of 1Gbps, while tenant B's traffic a 10Gbps link. If these are the only two tenants in the system, then the "natural" allocation would be $|\mathbf{R_A}| = 1$Gbps and $|\mathbf{R_B}| = 10$Gbps, respectively, which would violate the naive definition of network proportionality.

Formally, we define network proportionality as a generalization of max-min fairness. Let W_K be the weight associated to tenant K, e.g., $W_K = N_K$. Let F_K be the normalized allocation of tenant K, i.e., $F_K = |\mathbf{R_K}|/W_K$. Now, let $F^{ord}(\mathbf{R}) = \{F_{q_1}, \ldots, F_{q_m}\}$ be the sorted vector of normalized tenant allocations ($F_{q_i} \leq F_{q_{i+1}}$). The network proportional allocation $\mathbf{R}^*$ corresponds to the maximal normalized allocation $F^{ord,*}$ in increasing order, i.e., for any other feasible allocation F^{ord}, there exists q_i such that $F^{ord}_{q_i} < F^{ord,*}_{q_i}$ and $\forall q_j < q_i$, $F^{ord}_{q_j} = F^{ord,*}_{q_j}$. In essence, this allocation maximizes the minimum (normalized) tenant allocation.[2]

Next, we discuss the tradeoffs between these requirements.

2.3 Tradeoff Between Network Proportionality and Min-Guarantee

In this section we show that there is a tradeoff between achieving network proportionality and providing each VM a useful (i.e., large enough) bandwidth guarantee.

To illustrate this tradeoff, consider the example in Figure 2 showing two tenants A and B. A employs two VMs, while B employs eleven VMs. VMs A_1 and B_1 are hosted on the same machine; A_1 communicates with A_2, while B_1 communicates with the rest of the ten VMs of B. We assume that the access link of this machine is the only congested link in the system. According to the network proportionality requirement, A_1 should get $2/13$ of the access link

[1] While several providers (e.g., Amazon EC2) do not allow statistical multiplexing for the CPU and memory resources, others, such as RackSpace do [6].

[2] Previous work [22] has shown that the maximal lexicographical allocation is equivalent to max-min for multi-path routing, a problem to which network proportionality can be reduced to.

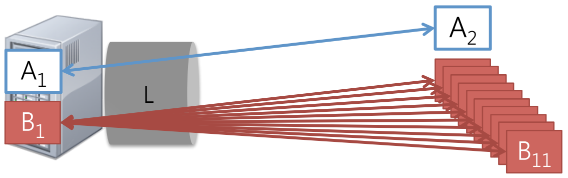


Figure 2: Network proportionality vs. min-guarantee. With proportionality, as B_1 communicates with more VMs, A_1's bandwidth can be decreased arbitrarily.

because A has two VMs and there are 13 VMs in total, while B_1 should get $11/13$. Unfortunately, one can arbitrarily reduce A_1's bandwidth by simply increasing the number of VMs that B_1 communicates with. While strictly speaking, proportionality still provides a min-guarantee, since the number of VMs in the data center is finite, the resulting guarantee is too low to be useful in practice.

If we consider the min-guarantee requirement alone, and assume no other VM can be admitted on the server, both A_1 and B_1 should each be guaranteed half of the capacity of the shared access link, just as each VM would be guaranteed half of the resources of the machine they are running on (recall that we are assuming identical VMs for now). This guarantee should not be affected by the traffic of other tenants in the network. However, as illustrated by our example, there is a hard tradeoff between network proportionality and min-guarantee: one can either achieve network proportionality or provide a useful min-guarantee, but not both!

2.4 Tradeoff Between Network Proportionality and High Utilization

We now show that even in the absence of the min-guarantee requirement, network proportionality is hard to achieve. In particular, we show that there is a tradeoff between network proportionality and high utilization.

To illustrate this tradeoff, consider the example in Figure 3 depicting two tenants A and B, each employing four VMs. Figure 3(a) shows a scenario in which their flows traverse the same congested link L of capacity C; each tenant gets $C/2$ of the aggregate bandwidth, and the network proportionality requirement is met. Now assume that VMs A_1 and A_3 start communicating along an uncongested path, P (Figure 3(b)). In order to preserve network proportionality, we need to *decrease* tenant A's allocation along link L. Unfortunately, if A deems its traffic along L more important than that between A_1 and A_3, A is disincentivized to use path P, which hurts network utilization.

We will refer to the ability of a tenant to use an uncongested path without being penalized on another (disjoint) congested path as the *utilization incentives* property. Thus, the tradeoff between network proportionality and high utilization can be reduced to a tradeoff between network proportionality and utilization incentives.

2.4.1 Congestion Proportionality

It might appear that one could get around this tradeoff by restricting the network proportionality requirement only to the traffic traversing congested links. We define *congestion proportionality* as network proportionality restricted to congested paths that involve more than one tenant. In other words, for each tenant K, congestion proportionality considers only $\sum_{i,j} R_K^{i,j}$, where there is at least a congested link along path $i \to j$ traversed by the flows of multiple tenants. For the example in Figure 3(b), since the path between A_1 and A_3 is used only by tenant A, it does not count towards A's use of congested resources when comparing the aggregate bandwidths of

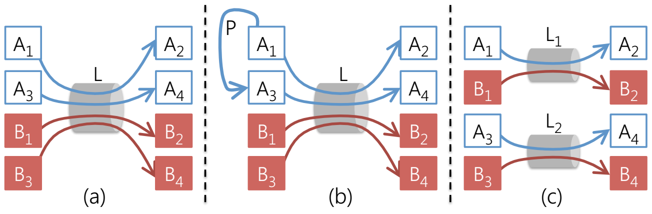


Figure 3: Network proportionality vs. high utilization. (a) VMs of tenants A and B have equal shares on link L. (b) If A_1 starts communicating with A_3, A's allocation on L decreases; thus, A may be incentivized to refrain from using the free path P. (c) Even with congestion proportionality, tenants can have incentives to keep links underutilized to increase their total allocation.

A and B. Thus, congestion proportionality does not disincentivize a tenant from using free resources.

Unfortunately, even congestion proportionality conflicts with high utilization, since it can incentivize tenants to artificially inflate or deflate their real demands. Consider Figure 3(c), where the traffic of tenants A and B is split across two congested links L_1 and L_2 with the same capacity C. Initially, assume that tenants have high demands and each tenant receives half of each congested link. This allocation trivially meets congestion proportionality.

Now assume the demand from B_3 to B_4 drops to a small value ϵ (all the other demands remain very high). As a result, $B_3 \to B_4$ will get ϵ on L_2, while $A_3 \to A_4$ will get $C - \epsilon$. In turn, congestion proportionality will change the allocation on L_1 so that $B_1 \to B_2$ gets $C - \epsilon$ and $A_1 \to A_2$ gets ϵ. By doing so, each tenant will still get same aggregate capacity C, and the system remains fully utilized.

However, congestion proportionality can still violate the utilization incentives property. For example, A may choose to send only $C - 2\epsilon$ on L_2. Since L_2 is no longer a congested link, congestion proportionality will only allocate L_1, by giving $C/2$ to both tenants. As a result, A ends up getting an aggregate allocation of $3C/2 - 2\epsilon$, while B will get only $C/2 + \epsilon$. By doing so, A increases her allocation, while the system utilization decreases from $2C$ to $2C - \epsilon$. This example also illustrates the fact that a tenant can artificially modify her demand to get a higher allocation. We refer to the ability to prevent such behavior as *strategy-proofness*.

2.4.2 Link Proportionality

We have shown that both network proportionality and its restricted version, congestion proportionality, end up compromising high utilization. To avoid this tradeoff, we further constrain the network proportionality requirement to *individual* links.

We define *link proportionality* as network proportionality restricted to a single link. More concretely, if the link is congested, link proportionality translates into max-min fairness between different tenants that communicate on that link.[3][4] A remaining question is what weight to associate with a tenant on a given link. The only constraint we impose for link proportionality is that the weight of any tenant K on a link L is the same for any communication pattern between K's VMs communicating over L and any distribution of the VMs as sources and destinations. For example, one can define the tenant K's weight on link L as the number of VMs of K that communicate on L. In another example, K's weight can be defined

[3]The max-min allocation is equivalent to the lexicographic maximum for both network proportionality and link proportionality.

[4]One can generalize link proportionality to the granularity of each VM instead of the tenant granularity, as we will discuss in §4.

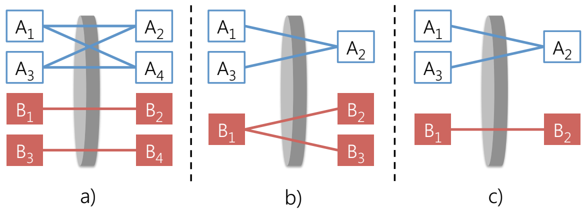


Figure 4: Two tenants sharing a single link using different communication patterns.

as N_K, i.e., total number of K's VMs in the network. Note that the latter allocation is similar to that of NetShare [19].

Since the allocation is independent across different links, link proportionality can achieve high utilization. However, link proportional allocations can be substantially different than network wide allocations such as those meeting congestion proportionality, since each VM can compete on a different number of congested links in the network.

More generally, we can classify all allocation policies to be applicable either at the link level or at the network level. We define link level allocations to have the *link-independence* property, meaning that the allocation at each congested link L is based only on the demands known at L and on static information about the VMs communicating on L (i.e., information that does not change during the lifetime of a VM). In the absence of link-independence, there can exist a congested link L whose bandwidth allocation can change due to a change in the allocation of another congested link L' or due to a change in the communication pattern of one of the VMs communicating on L. Network proportionality and congestion proportionality are not link-independent, while link proportionality is. All link-independent allocations provide incentives to use free resources.

Traditional allocation policies, such as per-flow fairness, are link-independent, and thus are capable of achieving high utilization. However, we next show that they cannot provide even link proportionality, nor do they provide min-guarantees.

2.5 Traditional Allocation Policies

The traditional approach to sharing the network is to apply *per-flow* fairness, where a flow is characterized by the standard five-tuple in packet headers. However, the Per-Flow mechanism can lead to unfair allocations at the VM granularity [11]. For instance, two VMs can increase the allocation between them at the expense of other VMs by simply instantiating more flows.

A natural "fix" is to use a *per source-destination pair* (Per-SD) allocation policy, 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 policy does not provide link proportionality either, because a VM that communicates with many VMs gets more bandwidth than a VM that communicates with fewer VMs. For example, a tenant that employs an all-to-all communication pattern between N VMs will get a bandwidth share of $O(N^2)$, while a tenant that performs one-to-one communication between the same number of N VMs will get a share of only $O(N)$. Figure 4 (a) shows one such example, where the bandwidth allocated to tenant A is twice that of tenant B.

To address this problem, previous solutions (e.g., Seawall [25]) have proposed using a *per source* (Per-Source) allocation policy. Per-Source assigns equal weights to all the sources communicating over a given link, and the bandwidth is divided accordingly. While this is fair to sources, it does not meet link proportionality since it is not fair to destinations. For example, in Figure 4(b), if VMs A_1, A_3, and B_1 are the sources, then the bandwidth allocation of tenant A would be twice that of tenant B. However, if the direction of the communication is reversed, tenant A would receive only half the bandwidth allocated to B. Therefore, there is a mismatch between the amount of traffic sourced and the amount received by a VM.

We refer to this mismatch between allocations in opposite directions as *asymmetry* and consider Per-Source to be asymmetric. Asymmetry is undesirable because it can result in application-level inefficiencies. For example, in a MapReduce setting, a VM hosting both a mapper and a reducer can experience a significant discrepancy between the bandwidth available for the mapper to send and for the reducer to receive, slowing down the application to the slowest component. More generally, VMs can experience large variations between the incoming or outgoing bandwidths, without showing a preference for one of them. A *per destination* (Per-Destination) allocation policy is asymmetric as well for similar reasons.

We have shown that both Per-Source and Per-Destination allocation policies fail to provide link proportionality. In addition, neither satisfies min-guarantee. Referring back to Figure 2, we can easily see that A_1's incoming bandwidth can arbitrarily be reduced by tenant B for Per-Source, and same applies to A_1's outgoing bandwidth for Per-Destination.

To summarize, we have established that there are fundamental tradeoffs between our requirements and that the traditional allocation policies are not satisfactory. We will next express these tradeoffs using a set of lower-level desirable properties. Based on the requirements and properties, we will propose new allocation policies to be implemented in cloud data centers.

Property	Description
P1. Work Conservation	Full utilization of bottleneck links.
P2. Strategy-Proofness	By being dishonest a tenant cannot improve her utility.
P3. Utilization incentives	Tenants are not disincentivized to use uncongested resources.
P4. Communication-Pattern Independence	The allocation does not favor some communication patterns compared to others.
P5. Symmetry	Reversing demands of all flows in the network does not change their allocations.

Table 2: Network sharing properties desirable for any bandwidth allocation policy in clouds.

3. NETWORK SHARING PROPERTIES

In this section, we describe a set of desirable properties that enable us to examine the above tradeoffs more explicitly. Table 2 summarizes these properties. We do not claim this to be a complete set of desirable properties, but rather a set that enables us to better understand the tradeoffs. Figure 5 captures the relationship between these properties, the requirements and tradeoffs discussed in §2.

***P1. Work conservation*:** As long as there is at least a tenant that has packets to send along link L, L cannot be idle. More formally, consider m tenants with demands $\mathbf{D} = \{\mathbf{D_1}, \ldots, \mathbf{D_m}\}$, and let P be an allocation policy that provides allocations $\mathbf{R} = \{\mathbf{R_1}, \ldots, \mathbf{R_m}\}$ (see Eq. 1). We say that P is work-conserving, iff for any flow of K that traverses an *uncongested* path $i \rightarrow j$, $R_K^{i,j} = D_K^{i,j}$. In other words, a link is either fully allocated, or it satisfies all demands. Surprisingly, unlike the case of a single resource, in a distributed setting, work conservation does not guarantee high utilization. The next two properties illustrate this point.

***P2. Strategy-proofness*:** Tenants cannot improve their allocations by lying about their demands [14]. Consider allocation policy P that provides allocation $\mathbf{R_K}$ to tenant K with demand $\mathbf{D_K}$. With each tenant K, we associate utility $U_K(\mathbf{R_K})$, a scalar function defined on K's

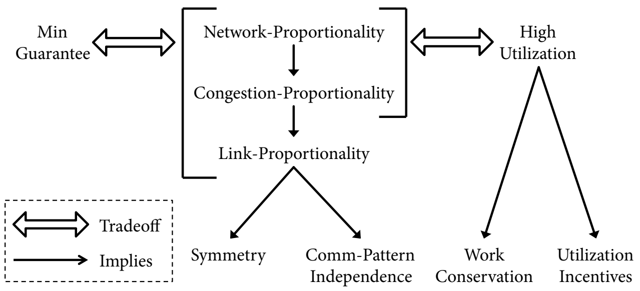


Figure 5: Requirements, properties, and tradeoffs between them.

allocation. We say that P is strategy-proof if tenant K cannot change its demands to get a better allocation with higher utility. That is, for any demand $\widehat{\mathbf{D_K}} \neq \mathbf{D_K}$, we have $U_K(\mathbf{R_K}) \geq U_K(\widehat{\mathbf{R_K}})$, where $\widehat{\mathbf{R_K}}$ is the allocation corresponding to $\widehat{\mathbf{D_K}}$. In other words, tenant K has no incentives to lie about her demands.

Unfortunately, any allocation policy that is unaware of the utility functions cannot satisfy the above definition. Moreover, even if we restrict the utility function of any tenant K to represent the total allocation of K, $U_K(\mathbf{R_K}) = |\mathbf{R_K}|$, i.e., each byte has the same utility for K, this property is still very challenging to achieve for any work conserving allocation.

For example, one can show that link proportionality fails to satisfy even this restricted version of strategy-proofness. Consider the example in Figure 6 consisting of two links with capacities C_1, and C_2, respectively. Assume a link proportionality policy, where each tenant has the same weight on each link, i.e., $A_1 \rightarrow A_2$, $A_3 \rightarrow A_4$ and $B_1 \rightarrow B_2$ have all equal weights. Now assume the demand from A_1 to A_2 is ϵ, while the other demands are infinite, and that $C_1 < C_2 < 2C_1$. In this case, only L_2 is congested. As a result, A gets an aggregate allocation of $\epsilon + \frac{C_2}{2}$ (i.e., ϵ on L_1 and $\frac{C_2}{2}$ on L_2), while B gets $\frac{C_2}{2}$. However, by artificially congesting L_1, A can reduce B's allocation to $\frac{C_1}{2}$, and *increase* her useful aggregate allocation to $\epsilon + C_2 - \frac{C_1}{2}$.

We believe that the above example can be extended to any work-conserving link-independent allocation. In fact, we are not aware of any work-conserving bandwidth allocation policy that is strategy-proof. We leave this challenge for future, and, instead, we next focus on a restricted version of this property, utilization incentives.

P3. Utilization incentives: Tenants are never incentivized to reduce their actual demands on uncongested paths or to artificially leave links underutilized. This property aims to preclude the scenarios described in §2.4, in which a tenant can improve her allocation, and, as a result, decrease the system utilization. More formally, let again $\mathbf{D_K}$ denote the true (real) demand of tenant K, and let $\widehat{\mathbf{D_K}}$ denote any demand, such that $D_K^{i,j} \geq \widehat{D_K^{i,j}}$, for any VM pair (i, j) for which the path $i \rightarrow j$ remains or becomes uncongested under demand $\widehat{D_K^{i,j}}$ ($i \rightarrow j$ can be either congested or not under $\mathbf{D_K}$), and $\widehat{D_K^{i,j}} = D_K^{i,j}$, otherwise. We say that an allocation satisfies the utilization incentives property, if it provides allocations $\mathbf{R_K}$ and $\widehat{\mathbf{R_K}}$, such that $U_K(\mathbf{R_K}) >= U_K(\widehat{\mathbf{R_K}})$, for any monotonic utility function of K.[5] In other words, tenant K will only reduce her utility by decreasing her real demand on an already uncongested link or on a link that becomes uncongested.

Note that the utilization incentives property is a particular case

[5]If a link L between $i \rightarrow j$ is congested for $D_K^{i,j}$ but not for $\widehat{D_K^{i,j}}$, we assume that the utility function is $U_K(\mathbf{R_K}) = |\mathbf{R_K}|$; for general utility functions it can be shown that this property is not achievable in this case by any work-conserving policy, since K can increase her most valued flows on link L.

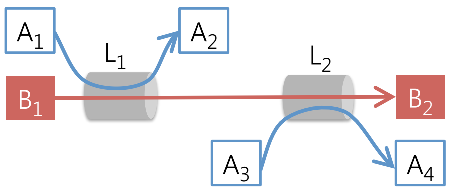


Figure 6: Link-independence vs. strategy-proofness: by increasing the $A_1 \rightarrow A_2$ traffic, A may increase her $A_3 \rightarrow A_4$ traffic.

of the strategy-proofness property, in which a tenant can only lie by reducing her demands on uncongested paths.

We have shown that network proportionality and congestion proportionality violate this property. Almost all link-independent allocations satisfy this property. Note that both work conservation and utilization incentives properties are necessary to satisfy the high utilization requirement.

P4. Communication-pattern independence: The allocation of a VM depends only on the VMs it communicates with and not on the communication pattern. Consider a set of source VMs, $\mathbb{Q}$ that communicates with a set of destination VMs, $\mathbb{P}$, where the VMs in $\mathbb{Q}$ and $\mathbb{P}$ do not communicate with any other VMs. Assuming sufficient demands, any communication pattern involving $\mathbb{Q}$ and $\mathbb{P}$ should result in the same aggregate allocation.

It is not always possible to achieve this property at the network level. For example, assume $\mathbb{Q} = \{A_1, A_3\}$, $\mathbb{P} = \{A_2, A_4\}$, and effective bandwidths along $A_1 \rightarrow A_2$ and $A_3 \rightarrow A_4$ to be much higher than that along $A_1 \rightarrow A_4$ and $A_3 \rightarrow A_2$ (e.g., due to different link capacities and/or background traffic). In this case, the first communication pattern ($A_1 \rightarrow A_2$, $A_3 \rightarrow A_4$) would receive a higher throughput than the second ($A_1 \rightarrow A_4$, $A_3 \rightarrow A_2$).

Consequently, we consider a simpler formulation of this property that is limited to a single congested link L. Some allocation policies still cannot support this property. In Figure 4(a), assume $\mathbb{Q} = \{A_1, A_3\}$, $\mathbb{P} = \{A_2, A_4\}$. Using Per-SD allocation, if A_1 and A_3 communicate with both A_2 and A_4, they get a larger share than a one-to-one communication such as $A_1 \rightarrow A_2$ and $A_3 \rightarrow A_4$. Per-Source allocation achieves this property since A_1 and A_3 will get $\frac{1}{2}$ of L's capacity regardless of the communication pattern (given sufficient demand).

We note that link proportionality implies communication-pattern independence; otherwise, allocations will not remain proportional when communication patterns change.

P5. Symmetry: If we switch the direction of all the flows in the network, then the reverse allocation of each flow should match its original (forward) allocation. More formally, assume G's routing is symmetric, and the capacities of every link are the same in both directions. If the demand of each tenant is transposed (i.e., the demand from i to j is equal to the original demand from j to i) and $\mathbf{R_K^T}$ is the resulting allocation for the transposed demands for tenant K, then $\mathbf{R_K^T} = \mathbf{R_K}\ \forall K$, where $\mathbf{R_K}$ is the original allocation.

Existing allocation policies make an implicit assumption as to whether the allocation is receiver- or sender-centric. However, it is difficult to anticipate application-level preferences. For example, server applications might value more the outgoing traffic while client applications might value more the incoming traffic. In the absence of application-specific information, we prefer allocations that provide equal weights to both incoming and outgoing traffic. As shown in §2, Per-Source and Per-Destination allocations do not provide the symmetry property. Proportionality requirements imply symmetry by definition, since the share of a tenant does not depend on the direction of its communication.

Figure 5 summarizes the desirable requirements, corresponding properties, and the tradeoffs discussed in §2 and §3.

4. PROPOSED ALLOCATION POLICIES

In the previous sections, we have described a set of requirements and properties desired of a network allocation policy and identified fundamental tradeoffs between them. In this section, we discuss how to navigate the tradeoffs shown in Figure 5 and describe three allocation policies that take different stands on these tradeoffs.

The first policy, PS-L, achieves link proportionality and can satisfy all the properties mentioned in §3 (except strategy-proofness). The second, PS-N, provides better proportionality at the network level (congestion proportionality in a restricted setting), but it does not fully provide utilization incentives. Finally, PS-P, provides minimum bandwidth guarantees in tree-based topologies (hence it does not provide proportionality). At the end of the section, we discuss how these policies can be implemented in practice as well as alternate pricing models.

Table 3 summarizes the properties achieved by these three policies as well as by the traditional network sharing policies.

Heterogeneous VMs: Before presenting the allocation policies, we first remove the assumption of all VMs being identical. We generalize to a model where tenants pay different flat-rate prices for individual VM's network share. 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 tenant's payment. Thus, each VM is also characterized by its network weight, in addition to its CPU and memory capacities. Intuitively, higher weights should result in higher bandwidth allocations. It is not difficult to extend link proportionality and min-guarantee to the heterogeneous model; for brevity we discuss these extensions as part of the presentation of our proposed allocation policies.

4.1 Proportional Sharing at Link-level

Proportional Sharing at Link-level (PS-L) is an allocation policy that provides link proportionality. The simplest way to understand PS-L is by considering a model in which each switch implements weighted fair queuing (WFQ) and has one queue for each tenant.

The weight of the queue for tenant A on link L is the sum of the weights of A's VMs that communicate through link L. For instance, let A's VMs have a communication pattern such that a set of VMs $\mathbb{Q}$ sends traffic to the set $\mathbb{P}$ over link L. Then, A's (unnormalized) weight will be $W_A = \sum_{X\in\mathbb{Q}} W_X + \sum_{Y\in\mathbb{P}} W_Y$. Consider Figure 4(b) and assume that all VMs have unit weights. Both tenants will be assigned a weight of 3, leading to an equal bandwidth distribution in both directions between them. In another example, on link L_1 in Figure 7, tenant A will have a weight of four since it has four VMs communicating and tenant B will have a weight of two.

One drawback of the above version of PS-L is that there is a simple strategy for tenants to increase their allocations. By sending an ϵ amount of data between all her VMs, a tenant can achieve a weight equal to her total number of VMs on any congested links. One fix is to simply use a weight for tenant A equal to the total weight of all of A's VMs, which can in fact be seen as an application of the NetShare model [19]. Another fix is to apply PS-L at a per VM granularity, rather than per tenant, as we describe next.

In this case, PS-L assigns to a communication between VMs X and Y on link L a weight of:

$$W_{X-Y} = \frac{W_X}{N_X} + \frac{W_Y}{N_Y}$$

where N_X is the number of other VMs X is communicating with on link L (similarly N_Y). For example in Figure 4(c), the per VM PS-L assigns weights of 1.5, 1.5, and 2 to $A_1 - A_2$, $A_3 - A_2$, and $B_1 - B_2$, respectively. So the flows between $A_1 - A_2$ and $A_3 - A_2$ would receive

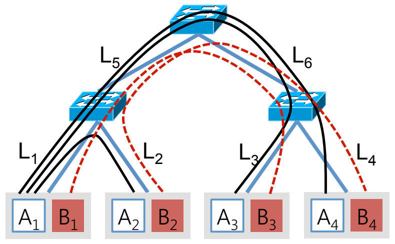


Figure 7: Example for illustrating the allocation policies.

$\frac{1.5}{5}$ of the link capacity, since A_2's weight is split across its two flows. This approach removes the incentives to send traffic between all of one VMs.[6]

As discussed in §2, link proportionality only offers a link level view of proportionality, which can be far from approximating proportionality at the network level. The next allocation policy aims to address this concern.

4.2 Proportional Sharing at Network-level

Proportional Sharing at Network-level (PS-N) is an allocation that aims to approximate congestion proportionality and can achieve it in a severely restricted setting. As a consequence, however, PS-N does not provide full utilization incentives (Table 3).

The intuition behind PS-N is that the communication between the VMs in a set $\mathbb{Q}$ has the *same total weight through the network* irrespective of the communication pattern between the VMs in $\mathbb{Q}$. This weight equals the sum of the weights of the VMs in $\mathbb{Q}$. For example, in Figure 7, the total weight of each tenant A or B is four regardless of the communication pattern between their VMs.

To achieve this, we extend PS-L to incorporate information regarding the global communication pattern of a tenant's VMs. PS-N uses a weight model that is similar to that of PS-L, as it sets the weight of the communication between VMs X and Y to be $W_{X-Y} = \frac{W_X}{N'_X} + \frac{W_Y}{N'_Y}$, with the difference that N'_X is the number of VMs that X communicates with across the *entire network* and not just over a particular link. For example, in Figure 7, PS-N would provide A_1-A_3 with a weight of $\frac{1}{4} + 1 = \frac{5}{4}$ on any link, while PS-L would provide it a weight of $\frac{1}{2} + 1 = \frac{3}{2}$ on link L_5, because A_1 communicates with only two VMs on L_5, but with four VMs in total.

PS-N strives to achieve proportionality in the absence of detailed knowledge about the load on each bottleneck link, assuming uniform load conditions throughout the network. Specifically, a set of VMs $\mathbb{Q}$ communicating using PS-N across a set $\mathbb{S}$ of bottleneck links achieves at least its proportional share of the total bottleneck capacity if the demand on each VM-to-VM communication is high enough and one of the following condition holds: (a) all bottleneck links have the same capacity and background weight (i.e., weight of traffic not involving VMs in $\mathbb{Q}$), or (b) all congested links are proportionally loaded, i.e., any two congested links of capacities C_1 and C_2 are loaded with total weights W_1 and W_2 such that $\frac{C_1}{C_2} = \frac{W_1}{W_2}$.[7]

For example, assume that the network shown in Figure 7 is fully provisioned, tenant A communicates all-to-all (i.e., each VM communicates with all other VMs of A), and tenant B communicates

[6] Note that the properties exhibited by the described per VM PS-L allocation are different for different demands. For example, in Figure 4(c), if the flow between A_1 and A_2 has a very small demand ϵ, the allocation between A_3-A_2 and B_1-B_2 will respect the ratio of $1.5 : 2$ instead of a $1 : 1$ ratio for a per VM proportionality. This can be addressed by taking into account the demands when assigning weights, but for brevity we do not detail that extension.

[7] There is a subtle difference between the two cases; for case (a), the weight does not include the VMs in $\mathbb{Q}$; for (b), it does.

	Per-Flow	Per-SD	Per-Source (Per-Destination)	Reservations (e.g., [10])	PS-L	PS-P	PS-N
Proportionality	×	×	×	×	link-proportionality	×	congestion-proportionality*
Min-Guarantee	×	×	×	√	×	√	×
Work Conservation	√	√	√	×	√	√	√
Utilization Incentives	√	√	√	√ †	√	√	×
Comm-Pattern Indep.	×	×	√	√	√	√	√
Symmetry	√	√	×	√	√	√	√

Table 3: Properties achieved by different network sharing policies. (*in a restricted setting, † also strategy-proof)

one-to-one (e.g., as shown in the figure). In this case, the two tenants achieve the same total bandwidth allocation (assuming high enough demands), since the congestion is on the access links, which have the same capacity and the same weight load. Indeed, the two tenants will have equal weights on the access links: $W_A = 8 \times \frac{1}{4} = 2$, since each VM will contribute with a weight of $\frac{1}{4}$ to each VM to VM flow, while $W_B = 4 \cdot \frac{1}{2}$. Note that PS-L would not provide network wide proportionality in this case, since A's weight would be twice that of B on the access links.

A drawback of PS-N is that each VM's weight is statically divided across the flows with other VMs, irrespective of the traffic demands. This further constraints the situations when PS-N achieves proportionality (requiring high demands on all flows), as well as makes PS-N lack the utilization incentives property. For example, assume that all the links in Figure 7 have the same capacity and L_5 is the only congested link. If A deems A_1's traffic to A_3 and A_4 more important than that to A_2, A may not send traffic between A_1 and A_2 to get a larger share for its traffic to A_3 and A_4 on L_5. We believe this could potentially be addressed if we include the demands when assigning weights, e.g., by not providing any weight to flows traveling uncongested paths. However, such a policy will be significantly more difficult to deploy, and we leave its exploration to future work.

4.3 Proportional Sharing on Proximate Links

The previous policies strive to achieve forms of proportionality and thus do not provide minimum bandwidth guarantees. To offer (useful) minimum bandwidth guarantees one would want to prioritize the allocation of a link based on the "importance" of that link with respect to the VMs using it. For example, on the access link of one host, we might want to allocate the link bandwidth based more on the weights of the VMs residing on that host and less on the weights of the remote VMs using the link.

Based on the above observation we generalize PS-L to derive the following weighting scheme: $W_{X-Y} = W_{Y-X} = \alpha \frac{W_X}{N_X} + \beta \frac{W_Y}{N_Y}$. The coefficients α and β allow different weights for VMs located on the two sides of the link. By setting specific values for α and β at different links in the network, one can use the generalized PS-L to achieve bandwidth guarantees for different network topologies.

In this paper, we present *Proportional Sharing on Proximate Links (PS-P)*, which is suitable for tree-based topologies (e.g., traditional data center architectures, VL2 [15], and multi-tree structures such as fat trees [7]). PS-P prioritizes VMs that are close to a given link. More precisely, PS-P uses $\alpha = 1$ and $\beta = 0$ for all links in the tree that are closer to X than Y, and $\alpha = 0$ and $\beta = 1$ for all links closer to Y than X.

In practice, PS-P translates into applying per-source fair sharing for the traffic towards the root of the tree and per-destination fair sharing for the traffic from the root. For example, on link L_1 in Figure 7, the three flows communicating with A_1 will share A_1's weight irrespective of the weights of A_2, A_3, and A_4, while on link L_6, A_1-A_3 will have A_3's weight and A_1-A_4, A_4's weight. These weights apply equally to both directions of the traffic.

PS-P provides absolute bandwidth guarantees for any VM to/from the root of the physical tree, since that VM competes on a given link to the root only with the other VMs in the same subtree. Obviously, cloud providers must deploy admission control to ensure that the total bandwidth guarantees of the VMs hosted within the subtree of any link L does not exceed L's capacity, similar to how CPU guarantees are offered on a given host. Guarantees are computed assuming all hosts are fully loaded with VMs.

For example, in Figure 7, on link L_1, B_1 competes only with A_1's weight irrespective of A_1's communication pattern. Thus, assuming all VMs have have equal weights and that the maximum weight on each host is 2, B_1 is always allocated at least $\frac{1}{2}$ of its access link capacity (L_1). Similarly, B_1 will get at least $\frac{1}{4}$ of L_5.

These guarantees can be used to offer different service models to tenants. The basic model offered by PS-P is similar to Oktopus's Virtual Cluster [10] (i.e., hose model), with the difference that PS-P is work conserving. The cloud provider can associate a minimum guaranteed bandwidth for every unit of VM weight and advertise it to customers through different VM configurations (CPU, memory and bandwidth). The guarantee is computed as the minimum share across all the layers of the tree. Tenants entirely collocated within a higher capacity subtree can achieve higher guarantees. The value of the guarantees can vary from VM to VM and from tenant to tenant.

PS-P can also expose a service model similar to Oktopus's virtual oversubscribed cluster (VOC) [10]. Specifically, the model can expose higher guarantees between each group of VMs collocated within a high-capacity subtree; the guarantees when communicating with VMs from a different group are scaled down with the oversubscription factor. Unfortunately, all models exposed by PS-P would have the same oversubscription ratio, that of the physical network. To fix this, PS-P can be applied within *virtual* topologies with different oversubscription characteristics. (The virtual topologies could themselves be build with a PS-P-like mechanism.)

We assume that if VMs can communicate via multiple paths, the routing protocol performs load balancing of the traffic across 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., [7, 8, 15, 23].

Similar to PS-L, PS-P can be implemented *per tenant*, which reduces the hardware requirements when PS-P is implemented at switches (§4.5). In this case, the weight of tenant A's queue through link L equals the number of VMs of A located in the subtree communicating through L. For example, in Figure 7, on link L_5, the weight of A will be $W_{A_1} + W_{A_2}$ (W_{A_1} is A_1's weight). The disadvantage of the per-tenant implementation is that the provided guarantees are not for each VM, i.e., when buying a VM, the VM is not guaranteed a minimum amount of traffic to/from the network core; rather, the tenant in aggregate is guaranteed that the bandwidth from all its VMs to/from the core equals the sum of the minimum guarantees that would be offered to each of its VMs by the per-VM PS-P. We believe that by selecting suitable values for α and β, one can generalize

PS-P to provide guarantees for other topologies, such as BCube [16] or DCell [18]. We leave this to future work.

4.4 Summary

Table 3 summarizes the properties achieved by the allocation policies presented in this section. All the described policies are work conserving, symmetric, and offer communication-pattern independence, but they make different choices for the rest of the properties. PS-L achieves link proportionality, does not provide guarantees, but does provide utilization incentives; PS-N achieves better proportionality at the network level, but does not provide full incentives for high utilization; lastly, PS-P provides guarantees and incentives for high utilization but not proportionality.

4.5 Deploying PS-L, PS-P and PS-N

We identify three deployment paths for the presented policies:

1. *Full switch support.* For this deployment, each switch must provide a number of queues equal to or greater than the number of tenants communicating through it, and must support weighted fair queuing. All the described policies can be implemented with such switch support.

2. *Partial switch support.* PS-N and PS-P are amenable to implementation using CSFQ [26], which does not require support for per-tenant or per-VM queues at switches.

3. *No switch support.* There are two types of hypervisor-only implementations. First, a centralized controller can enforce rate-limiters at hypervisors based on the current network traffic and implement any of the proposed policies. Second, PS-N (and we believe PS-P as well) could be implemented through a distributed mechanism similar to Seawall [25].

In this paper, we focus on the full and partial switch support deployments and only sketch the hypervisor-only deployment, leaving it to future work. We present our evaluation in §5 and we present more details related to practical deployment issues in §6.

4.6 Other Models

In addition to the flat-rate per VM pricing model, PS-P can also accommodate a per-byte pricing model, such as paying for the bandwidth above the minimum guarantees (like DRP [9]), or simply paying for all bytes. We also note that if the per-byte price is constant across tenants and traffic volume, proportionality still is a desirable property. However, congestion pricing or other forms of price discrimination can offer alternative ways of sharing the network, which would not benefit from proportionality.

Finally, we note that it is possible to create a more complex allocation policy that provides guarantees and shares proportionally *only* the bandwidth unused by guarantees. In a nutshell, this allocation would work as follows. Assume the total weight of all the VMs in the subtree delimited by a link L (for which we provide guarantees on L) is W_G and the weight of the VMs currently active in the subtree is W_C. We note by $W_P = W_G - W_C$ as the weight to be divided proportionally (of the VMs not active). In this context, the weight of tenant A on link L is $W_A = W_{A_S} + WP \times \frac{W_{A_T}}{W_T}$, where W_{A_S} is the weight of A's VMs in the subtree, while W_{A_T} and W_T are the weights of all A's VMs, and all VMs, respectively, active on L.

5. EVALUATION

In this section we evaluate the allocation policies presented in §4. We divide the results in three parts.

- First, we consider a single congested link and explore the behavior of these policies under various scenarios (§5.1).

- Second, we extend our experiments the network level, exemplify the aforementioned tradeoffs using suitable small-scale examples, and observe how these policies behave (§5.2).

- Finally, we leverage traces obtained from a 3200-node production cluster at Facebook to validate the properties exhibited by these policies on a large scale (§5.3).

We generated the results using a flow-level simulator written in Java and validated the simulation results by experiments in the DETERlab testbed [4]. For this purpose, we implemented switch support for PS-L, PS-P and PS-N using per-flow queues in a software router implemented in Click [20]. The implementation is using kernel-mode Click and consists of $\sim$500 lines of C++ code for the new elements and $\sim$2000 lines of python scripts to generate configuration files. We also developed support for PS-N with CSFQ with additional $\sim$400 lines of C++ for the Click elements. To achieve high performance, we deploy Click in kernel mode and approximate CSFQ using fixed point operations. Unless otherwise specified, the presented results are obtained in simulation.

Since Per-Flow's allocation can be arbitrary, depending on the number of flows, we only consider a maximum of one flow between any source-destination pair. In this case, Per-SD and Per-Flow are equivalent, and therefore, we omit Per-SD from the results. For ease of exposition, we assume all VMs to have a unit weight. The highlights of our findings are as follows:

- The tradeoff between proportionality and bandwidth guarantee is also evident at the network level. While PS-P is able to provide the highest guarantee, PS-N exhibits maximum proportionality in network allocation.

- The relative behaviors of these policies scale to large-scale clusters. PS-N is very close to achieving network proportionality for the small MapReduce jobs, and both PS-P and PS-N improve the minimum bandwidth allocations of the tasks of individual jobs and therefore result in an approximate speed-up of $\sim 15\times$ in the shuffle time of the small jobs.[8]

5.1 Link-Level Scenarios

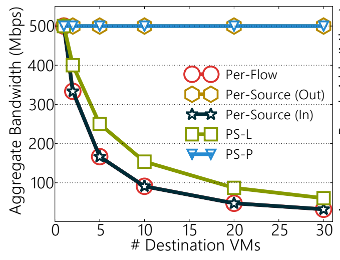
For the first set of experiments, we focus on a single congested link and evaluate multiple allocation policies by varying workloads. Since PS-N would provide the same allocation as PS-L on a single link (assuming that the VMs in our experiments only communicate over that congested link), we omit it from the following discussion.

Figure 8 depicts a scenario similar to the one in Figure 2, where two VMs, A_1 and B_1, are collocated on the same physical host; A_1 communicates with VM A_2, while B_1 communicates with N other VMs. We assume VMs have high demands and the link capacity is 1Gbps. Figure 8 plots the bandwidth allocation of tenant A for increasing values of N. Figure 8(a) presents simulation results, while Figure 8(b) presents actual allocations achieved on the testbed. Remember that Per-Source is asymmetric, and so we present both incoming and outgoing allocations for it.

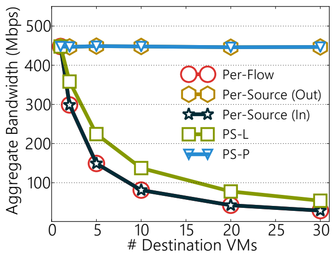
We make four observations from these results.

- First, PS-P maintains the same throughput for tenant A regardless of N. This is because A_1 is guaranteed a minimum bandwidth of half its access link capacity; the same is true for the outgoing bandwidth allocated by Per-Source.

[8]During the shuffle phase, mappers of a MapReduce job transfer intermediate data to the reducers over the network.

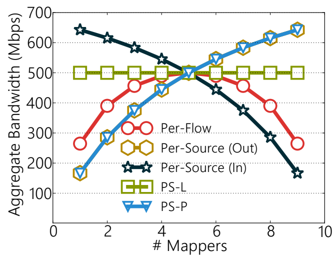


(a) Simulation results

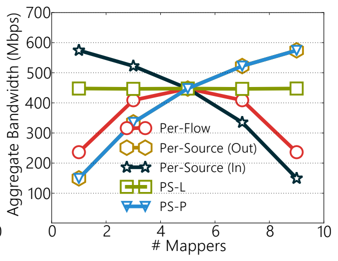


(b) Experimental results

Figure 8: Link level bandwidth allocation of tenant A, while tenant B has an increasingly larger number of VMs (similar to the scenario in Figure 2). PS-P maintains a constant throughput for tenant A.



(a) Simulation results



(b) Experimental results

Figure 9: Network allocation of tenant B on a congested link, while tenant A varies the number of mappers (senders). Both tenants use the many-to-many communication pattern commonly found in MapReduce shuffles.

- Second, PS-L provides tenant A its proportional share.
- Third, Per-Flow and Per-Source policies provide neither proportionality nor guarantees.
- Lastly, our implementation matches the simulation results, modulo the fact that the link capacities available on DETERlab are smaller (the available bandwidth is ~900Mbps instead of advertised 1Gbps).

In Figure 9, we consider two MapReduce jobs A and B (jobs play the role of tenants in this discussion), each with 10 VMs. While job B has 5 mappers and 5 reducers communicating over the congested link under consideration, we vary the ratio between the number of mappers (M) and reducers (R) of job A while keeping $(M+R) = 10$. As before, we show both simulation and testbed results.

We observe that PS-L achieves a proportional allocation that is not affected by the change in M, while PS-P's allocation increases as $\frac{M}{M+5}$ (5 is the number of mappers of tenant B). We also notice that Per-Flow achieves proportionality only when the distribution of mappers and receivers is the same between the two jobs (at $M = 5$), since only at this point both the jobs have equal number of flows; remember that the number of flows is $M \times R$.

5.2 Network-Level Scenarios

We now turn our attention to the network-wide allocation under different scenarios. For these experiments we include PS-N, because its network-wide allocations will be different than those of PS-L.

We start off with the simple scenario illustrated in Figure 10(a). In this particular example, we have a small tree with eight servers and two tenants A and B, each with one VM in each of the servers.

Strategy	Full Bisection BW			4× Oversubscribed		
	Sim	Exp		Sim	Exp	
Per-Flow	7.00	6.99		18.98	18.91	
Per-Source	1.00	1.00		6.99	6.75	
PS-L	4.00	4.03		7.00	7.20	
PS-P	1.00	0.99		7.00	7.13	
PS-N	1.00	WFQ	CSFQ	5.29	WFQ	CSFQ
		1.00	1.15		5.30	5.55

Table 4: Simulation vs. Experimental Results (Ratio of Aggregate Bandwidth of the tenants, B/A)

We assume that tenant A communicates using a pairwise one-to-one communication pattern between its VMs (i.e., $A_i \leftrightarrow A_{i+4}$, where $i = \{1, \ldots 4\}$), while tenant B communicates all-to-all (i.e., B_i communicates with all B_j, where $j \neq i$). For simplicity, we assume all the VMs have equal weights and infinite bidirectional demands. From the network proportionality point of view, an ideal allocation would provide both tenants the same bandwidth.

Figure 10(b) presents the bandwidth allocation for the two tenants in Figure 10(a) when the core is fully provisioned (i.e., the network has full bisection bandwidth). In this case, the access links are the congestion points. We see from Figure 10(b) that PS-P, PS-N and Per-Source policies are able to match the desirable equal allocation. However, PS-L provides B twice as much bandwidth as A on each access link, since B competes with four VMs on each access link against only two VMs of A—we use the version of PS-L proportional the number of active tenant VMs on the link. Consequently, at the network level, Per-Flow and PS-L favor dense communication patterns such as all-to-all over sparse communication patterns.

Figure 10(c) presents the allocations for tenants A and B when the core is under-provisioned by a factor of 4× (each of the aggregation and core layers being oversubscribed by 2×). Given our setup, core links are the bottlenecks for tenant A. In this case, PS-P, PS-L and Per-Source policies allocate the core bandwidth equally between the two tenants, while allowing tenant B to fully utilize the aggregation-level bandwidth for free. However, PS-N penalizes tenant B for utilizing the aggregation-level bandwidth because half the weights of each of the tenant B's VMs is utilized in aggregation-level communication, and it provides twice as much core bandwidth to tenant A. Thus, PS-N provides the best proportionality at the network level (see Table 4 for quantitative results).

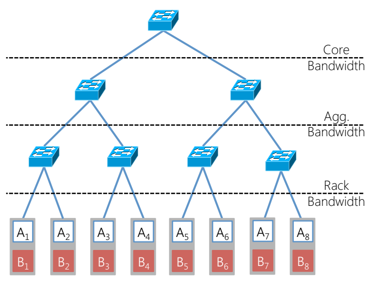
We validate the simulation results presented in the above two cases through experiments on a testbed with the same topology. Table 4 presents the comparison between the simulation and the experimental results. The experimental results closely match those from the simulation, with an error margin lower than than 4%.

In the next two experiments, we focus on highlighting the tradeoff between proportionality and bandwidth guarantees at the network level. First, consider a scenario, where two tenants A and B have the same number of VMs (64) deployed on a fully provisioned tree network —we use a 32 server cluster, each server containing one VM of each tenant. VMs of tenant A communicate in a (random) one-to-one fashion. B runs a MapReduce job, and his VMs are divided in two sets M and R, such that all VMs in M communicate to all in R. For ease of exposition, we consider the communications to be bidirectional.[9]

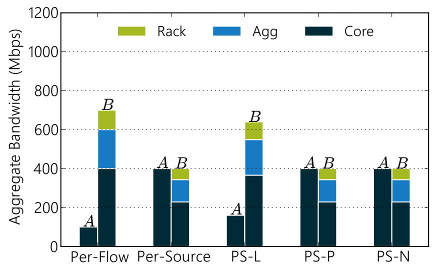
Figure 11 presents the ratio between the aggregate bandwidths of B and A for PS-P and PS-N in this setting.[10] The fully proportional

[9] Otherwise, congestion will not appear uniformly, and the results will become more difficult to understand.

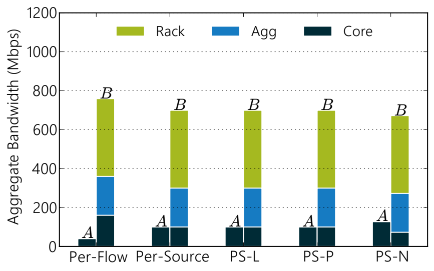
[10] We did not include the other policies since they would offset the chart, making it significantly harder to visualize.



(a) Experiment



(b) Full bisection bandwidth network



(c) 4× Oversubscribed network

Figure 10: Simple scenario with two tenants having equal number of VMs and a uniform deployment. Tenant A communicates using a pairwise one-to-one communication pattern and B all-to-all. (b) PS-P, PS-L and Per-Source policies achieve perfect proportionality in presence of full bisection bandwidth; (c) PS-N provides the best allocation ratio among the compared policies when the network is oversubscribed.

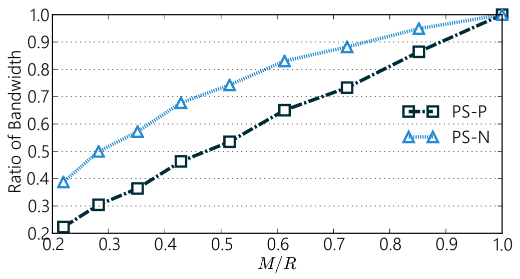


Figure 11: PS-N is more proportional than PS-P.

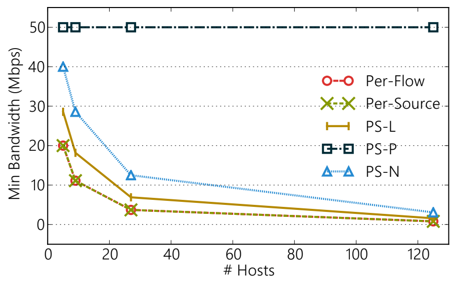


Figure 12: PS-P gives the highest bandwidth guarantee.

allocation would allocate equal network shares to both A and B, and would appear as a horizontal line at $y = 1$. However, this allocation is not always possible, since the MapReduce communication pattern is limited by $\min(M, R) \times C$, where C is the access link capacity. Figure 11 shows that PS-P provides direct proportionality between $\min(M, R)$ and tenant B's bandwidth: each VM is guaranteed its fair share of the network up to the core, and MapReduce will communicate with an aggregate throughput of $\min(M, R) \times Bw$, where Bw is the per-VM bandwidth guarantee ($Bw = \frac{C}{2}$ in this case). PS-N, on the other side, provides better proportionality than PS-P, i.e., the ratio of allocated bandwidths is closer to 1.0.

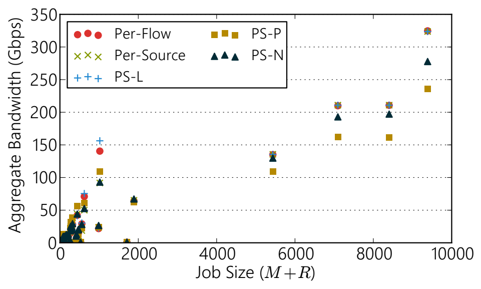
Proportionality, however, comes at a cost as expected from the tradeoffs presented in §2; the minimum bandwidth allocation of a VM provided by PS-N can become arbitrarily small in practice. Figure 12 presents the minimum guarantee provided by PS-N and PS-P for a scenario similar to the one shown in Figure 7 but with a slightly different communication pattern. In addition to the *N-to-one* communication pattern employed by A (i.e., all of A's VMs send to A_1), each of the N VMs sending traffic to A_1 also communicates with another of the N VMs, so that they can utilize the rest of the links. Without this additional flow from each sender, the throughput of A's VMs would be limited by the traffic going into A_1, and the minimum bandwidth would be affected by the hose model rather than the allocation scheme. We study the variation of minimum of the bandwidth allocations in such a scenario with increasing network size. Figure 12 shows that PS-P can maintain the same minimum guarantee regardless of the network size (for the same oversubscription ratio), while PS-N's minimum bandwidth decreases with the network size given this "bad" communication pattern.

5.3 Trace-driven Simulations

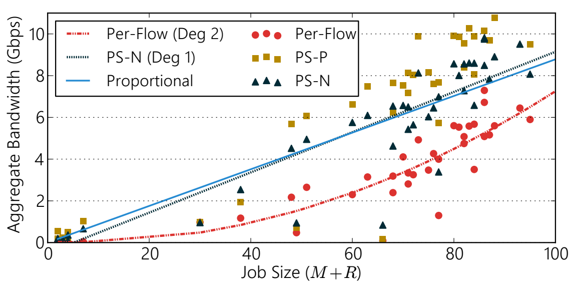
So far we have evaluated the proposed and existing network allocation policies in synthetic, small-scale scenarios. To understand their large-scale implications, we performed some experiments using MapReduce traces from a 3200-node Facebook production data center [12]. Our goal is to identify the network shares that different MapReduce jobs would achieve given different network allocation techniques.

We consider a one hour window in the trace and observe the number of jobs involved in active shuffle at a minute's interval in that hour. In our trace, an average number of 73 jobs were in active shuffle at any given time, with a maximum of 100 jobs and a minimum of 49. We then create a snapshot in time at the point when most jobs are in active shuffle, generate corresponding traffic matrices and observe the allocation of different policies. We infer the network position of each mapper and reducer based on the IP addresses and the names in the log files. The inference shows a heterogeneous cluster with an average of four active tasks per server, with some servers having up to 12 tasks. Due to the lack of knowledge about the oversubscription factor, we consider both full bisection bandwidth and 4× oversubscribed topologies with 20 aggregate switches and 160 racks. Note that the trace does not contain information about the processing times of each job; hence, we do not try to emulate entire running times of the jobs.

Figure 13(a) presents the network allocations for all the active jobs based on their size (measured by the number of mapper and reducer tasks) for the full bisection bandwidth case. The total network utilization was the same since all evaluated allocations are work conserving disciplines, but the distribution of bandwidth among jobs is different. During the shuffle phase, a MapReduce job with M mappers and R reducers can launch $M \times R$ flows, one from each of the mappers to each of the reducers. In such a scenario, a Per-Flow allocation can potentially give a network share proportional to $M \times R$ to a job whose payment was proportional to $M + R$ (the number of map and reduce slots it was allotted). Thus network allocations can be quadratic making allocations significantly unfair to small jobs.



(a) All Jobs



(b) Small Jobs Only

Figure 13: Network allocation of MapReduce jobs on a 3200-node Facebook production cluster under different policies.

Strategy	Full Bisection BW		4× Oversubscribed	
	RMSD	Speed-up	RMSD	Speed-up
Per-Flow	1.76	1.00	2.60	1.00
Per-Source	1.22	9.64	1.31	11.91
PS-L	1.72	2.49	2.39	10.73
PS-P	1.61	16.61	1.19	14.37
PS-N	1.00	9.96	1.00	11.77

Table 5: Normalized deviation from proportional allocation measured using RMSD and speed-up in the shuffle completion time for small jobs.

To better understand this unfairness against small jobs, we consider jobs with $(M+R) < 100$, that form a substantial fraction of the total jobs. Figure 13(b) shows the manifestation of this quadratic allocation problem with Per-Flow, which gives considerably lower shares to the small jobs. We also note that PS-N almost closely matches the proportionality line$-(M+R)\frac{BW}{N}$, where BW is the total bandwidth and N is the total number of VMs of this data subset.

Table 5 presents quantitative results for small jobs for: (i) proportionality and (ii) speed-up in the shuffle completion time compared to Per-Flow, for both fully subscribed and oversubscribed topologies. We quantify proportionality by using the Root-Mean-Squared Deviation (RMSD) of the allocations from the proportional allocation and normalize them w.r.t. PS-N (which exhibits the least deviation for both topologies). The shuffle completion time of a job is bottlenecked by the last finishing flow [12], which in turn is dictated by the task having the minimum bandwidth allocation across all the tasks in a job. We thus approximate the shuffle time by dividing the bytes transferred per task by the bandwidth of the slowest transferring task. We report the median speed-up value relative to the Per-Flow (TCP) case which performs the worst.[11] PS-P performs the best (since it maximizes the minimum bandwidth guarantee for a task) with speed-ups of ~15× for both topologies. Per-Source and PS-N also give significant improvements for the small jobs.

6. PRACTICAL CONSIDERATIONS

In this section, we present more details for the practical challenges towards deploying the allocation policies described in §4.

Full Switch Support: Today, there are switches that already have hardware capabilities matching our requirements (one queue per tenant, weighted fair queueing) such as routers with per-flow WFQ support [2], however most data center switches today have 8-64 hardware queues [5]. Determining the cost of this extra support compared to today's data center switches is a place for future work.

We also note that the weights for the tenant queues must be updated in time. For PS-P, weighs on a link L must be updated only when a new VM is started in the subtree delimited by L. For PS-L and PS-N the weight of tenant A through link L needs to be updated when there is a new pair of VMs belonging to A communicating on L. Updating weights can be done by (1) a centralized controller or (2) through the data-plane packets, which need to contain the tenant ID and weight information in packets. Providers can implicitly encode tenant IDs into different sets of IP or (virtual) MAC address ranges (e.g., using something like NetLord [21]). The weight of a VM can be encoded in packets through the use of the QoS bits in the IP header, filled out by hypervisors (for security reasons), which allows for 256 weights.

To support PS-L, switches require no additional information. For PS-P, switches need to be configured with one bit for each interface, identifying whether the interface is facing hosts or facing the network core.

Deploying PS-N is the most challenging, since it requires coordination between the source and the destination hypervisors for setting the weight of a source-destination pair. This weight must also be communicated to switches. Note that all the flows between a source and a destination contain the same weight. For the first packet between two VMs, the source's hypervisor inserts its allocated weight and the destination adds its weight to the returned packets. The subsequent packets contain the full weight of the source destination pair. When one of the endpoints starts communicating with another VM, its hypervisor updates the weights of the ongoing flows to the other endpoints. For practical purposes, the presence or absence of a communication between two VMs can be identified by using a threshold for the outgoing/incoming rate. Note that the 8 QoS bits in the IP packet header may offer too little weight granularity for PS-N. One approach to increase the number of available bits is to use encapsulation, e.g., something like NetLord [21]. In this case, only the MAC addresses are used for switching from the encapsulated header and many more other bits can be used for this purpose. VLAN tags can also be used to carry weights.

Partial Switch Support: CSFQ [26] was proposed to implement weighted fair queueing between flows without requiring per-flow state in the core switches but only in the edge switches. In data centers, we can use CSFQ and only maintain per-VM (or even per-tenant) state inside hypervisors at end points, and no such state in switches. PS-N can directly be implemented using vanilla CSFQ. PS-P can also be implemented using CSFQ but requires a slight change in the mechanism. In particular, each flow needs to contain two weights, the source and the destination weights, and switches

[11]The mean has a similar behavior albeit a few outliers with large speed-ups.

need to swap the two weights when the direction of packets changes from flowing towards the core to flowing away from the core (towards servers). However, a CSFQ-based deployment does require CSFQ support in switches. While we do believe hardware support for CSFQ can be inexpensive and fast, this claim remains to be proven by hardware manufacturers.

No Switch Support: TCP is known to provide per-flow fairness. If we use a single flow between a source and a destination or we impose an aggregate equivalent behavior in hypervisors, we achieve per source-destination fairness. If we are able to use weights between these flows (e.g., two TCP flows weight twice as much as a single flow and get twice as much bandwidth on a congested link) we can effectively implement PS-N. Seawall [25] aims to achieve this purpose. Thus, we expect one would be able to use Seawall, and instead of using per-source weights, use the weights given by PS-N. Weighted Flow Assignment (WFA) [12] can also approximate PS-N shares at the application layer using multiple TCP flows. We believe PS-P could be approximated with similar (but more complex) mechanisms.

7. RELATED WORK

Recently, there have been a few proposals for sharing cloud networks. Seawall [25] proposes a hypervisor-based mechanism for enforcing a generalized TCP-like behavior between VMs, where each TCP-like flow can have an arbitrary weight (rather than a single weight as in the case of TCP). Using this mechanism Seawall implements a per-source allocation policy. Therefore, Seawall is mostly orthogonal to our paper; in fact, Seawall's mechanism may be used to implement PS-N and PS-P. We leave this as future work.

Oktopus [10] and SecondNet [17] propose static reservations throughout the network to implement bandwidth guarantees for the hose model and pipe model, respectively. The main drawback of reservation systems is that they do not achieve the work conservation property, since the unused bandwidth is not shared between tenants. On the other hand, the advantage of reservation systems is that they can achieve more complex virtual topologies regardless of the physical location of the VMs. PS-P can support different bandwidth guarantees for different tenants by using carefully selected weights, but cannot support virtual topologies that are different than the physical topologies. For this purpose, reservation systems could be combined with our proposed allocation policies, which can be applied within each reserved virtual topology.

Gatekeeper [24] proposes a per-VM hose model with work conservation. Gatekeeper uses a hypervisor-based mechanism, which, however, works only for full bisection-bandwidth networks. In this paper we have described the PS-P allocation policy which supports a similar model for arbitrary tree networks, and described possible deployments using switch support; we are currently investigating how to implement PS-P using only hypervisors as well.

NetShare [19] advocates network sharing through the use of per-tenant weights that are constant throughout the network. This model can be used to implement a form of link proportionality.

Congestion Exposure (ConEx) [3] is a recent IETF effort that aims to equalize the number of dropped packets (congestion-volume) of different entities. By applying the ConEx mechanism between VMs one could achieve a Per-SD allocation. By applying ConEx between tenants it appears that the closest abstraction achieved is some form of congestion proportionality (but which also considers links congested by a single tenant). However, the precise set of properties of this approach remain to be determined.

8. CONCLUSIONS

In this paper, we have focused on understanding and exploring several key requirements and properties for network allocation in data centers. In summary, we have identified three main requirements: min-guarantee, proportionality (ranging from the network level to the link level) and high utilization, and a set of properties to guide the design of allocation policies in the tradeoff space.

In addition, we have introduced three allocation policies—PS-L, PS-P and PS-N—to navigate the tradeoff space. We have evaluated the proposed allocation policies using simulation and a software switch implementation. Through hand-crafted examples and traces of MapReduce jobs from a production cluster, we have shown that they achieve their intended properties. However, much more remains to be done. The allocation policies we have proposed in this paper should be seen as merely starting points in exploring the tradeoff space.

Acknowledgments: We thank the anonymous reviewers and our shepherd for their valuable feedback. Part of this work was supported by the NSF Award CNS-1038695.

9. REFERENCES

[1] Amazon web services. http://aws.amazon.com.
[2] Cisco 7500 series. http://goo.gl/m0Ve0.
[3] Congestion Exposure. http://datatracker.ietf.org/wg/conex/.
[4] DETERlab. http://www.isi.deterlab.net.
[5] HP 5900 ToR switch. http://goo.gl/kcycc.
[6] Rackspace Cloud Servers vs. VPS Platforms. http://goo.gl/LPxIJ.
[7] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *SIGCOMM*. ACM, 2008.
[8] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat. Hedera: Dynamic Flow Scheduling for Data Center Networks. In *NSDI*, 2010.
[9] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. The price is right: Towards location-independent costs in datacenters. In *Hotnets*, 2011.
[10] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards Predictable Datacenter Networks. In *ACM SIGCOMM*, 2011.
[11] B. Briscoe. Flow rate fairness: Dismantling a religion. *ACM SIGCOMM Computer Communication Review*, 2007.
[12] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica. Managing data transfers in computer clusters with Orchestra. In *SIGCOMM*, 2011.
[13] 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.
[14] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica. Dominant resource fairness: fair allocation of multiple resource types. In *USENIX NSDI*, 2011.
[15] 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.
[16] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers. *ACM SIGCOMM*, 2009.
[17] 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 *CoNEXT*. ACM, 2010.
[18] 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.
[19] T. Lam, S. Radhakrishnan, A. Vahdat, and G. Varghese. NetShare: Virtualizing Data Center Networks across Services. *Technical Report, UCSD*, 2010.
[20] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek. The click modular router. *SIGOPS Oper. Syst. Rev.*, 33(5):217–231, 1999.
[21] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary. Netlord: a scalable multi-tenant network architecture for virtualized datacenters. In *ACM SIGCOMM*, 2011.
[22] B. Radunović and J.-Y. L. Boudec. A unified framework for max-min and min-max fairness with applications. *IEEE/ACM Trans. Netw.*, Oct. 2007.
[23] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving Datacenter Performance and Robustness with Multipath TCP. In *ACM SIGCOMM*, 2011.
[24] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. Guedes. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. In *USENIX WIOV*, 2011.
[25] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha. Sharing the Data Center Network. In *Usenix NSDI*, 2011.
[26] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: achieving approx. fair bandwidth allocations in high speed networks. In *SIGCOMM'98*.