Networking 2013 1569698241

Trust-based Grouping for Cloud Datacenters: improving security in shared infrastructures

Daniel Stefani Marcon, Rodrigo Ruas Oliveira, Miguel Cardoso Neves,
Luciana Salete Buriol, Luciano Paschoal Gaspary, Marinho Pilla Barcellos
Institute of Informatics – Federal University of Rio Grande do Sul
Av. Bento Gonçalves, 9500 – 91.501-970 – Porto Alegre, RS, Brazil
Email: {daniel.stefani, ruas.oliveira, mcneves, buriol, paschoal, marinho}@inf.ufrgs.br

***Abstract*—Cloud computing can offer virtually unlimited resources without any upfront capital investment through a pay-per-use pricing model. However, the shared nature of multi-tenant cloud datacenter networks enables unfair or malicious use of the intra-cloud network by tenants, allowing attacks against the privacy and integrity of data and the availability of resources. In this paper, we introduce a resource allocation strategy that increases the security of network resource sharing among tenant applications. The key idea behind the strategy is to group applications of mutually trusting users into virtual infrastructures (logically isolated domains composed of a set of virtual machines as well as the virtual network interconnecting them). This provides some level of isolation and higher security. However, the use of groups may lead to fragmentation and negatively affect resource utilization. We study the associated trade-off and feasibility of the proposed approach. Evaluation results show the benefits of our strategy, which is able to offer better network resource protection against attacks with low extra cost.**

I. Introduction

Cloud Computing has become the platform of choice for the delivery and consumption of IT resources. It offers several advantages, such as pay-per-use pricing model, on-demand self-service, broad network access and rapid elasticity. In this model, providers avoid allocating physically isolated resources for each tenant. Instead, they implement cloud datacenters as highly multiplexed shared environments, with different applications coexisting on the same set of physical resources [1]. Therefore, they can increase resource utilization, reduce operational costs and, thus, achieve economies of scale.

Providers, however, lack mechanisms to capture and control network requirements of the interactions among allocated virtual machines (VMs) [2]–[4]. For example, congestion control mechanisms used in intra-cloud networks (including TFRC [5] and DCCP [6]) do not ensure robust traffic isolation among different applications, especially with distinct bandwidth requirements [7]. Thus, communication between VMs of the same application takes place in an uncontrolled environment, shared by all tenants. This enables selfish and malicious use of the network, allowing tenants to perform several kinds of attacks [1], [8], [9]. Selfish attacks are characterized by the consumption of an unfair share of the network (i.e., performance interference attacks [1]), while malicious ones include man-in-the-middle, extraction of confidential information from tenants and denial of service (DoS).

Such attacks hurt both tenants and providers. Tenants have weaker security guarantees and unpredictable costs (due to potential attacks against network availability), since the total application execution time in the cloud is influenced by the network [3]. Providers, in turn, lose revenue, because attacks can affect network availability and reduce datacenter throughput [10].

Vulnerabilities of cloud network resource sharing have been studied in [8], [9]. Recent proposals try to increase network security through network-aware resource allocation strategies [1], [11], [12]. Nonetheless, these schemes can neither protect the network from malicious insiders nor prevent selfish behavior by other applications running in the cloud environment.

In this paper, we propose a resource allocation strategy for Infrastructure as a Service (IaaS) providers. Our approach increases the security of network resource sharing among tenant applications by mitigating selfish and malicious behavior in the intra-cloud network. Unlike previous work, we investigate a strategy based on grouping of applications in virtual infrastructures[1] (VIs).

Grouping applications into VIs has two benefits, as follows. The first one is related to security: grouping can provide isolation among applications from mutually untrusted tenants. That is, the system becomes more resilient against tenants that would try to cause disruption in the network, capture confidential information from other applications or use a disproportionate share of resources. The second benefit regards performance, since grouping allows cloud platforms to provide performance isolation among applications with distinct bandwidth requirements. In summary, security and performance isolation increase network guarantees for applications and reduce tenant cost. Moreover, the proposed approach does not require any new hardware. In fact, it can be deployed either by configuring network devices or by modifying VM hypervisors, similarly to [3], [10].

On the other hand, the number of groups created is pragmatically limited by the overhead of the virtualization technology. Moreover, groups may lead to internal resource fragmentation while allocating requests and negatively affect resource utilization. Therefore, we study different strategies to group tenants based on their mutual trust and on the network requirements (bandwidth) of their applications.

Overall, the main contributions of this paper are summarized as follows:

[1]The term virtual infrastructure is used to represent a set of virtual machines as well as the virtual network interconnecting them. This concept is formally defined in Section IV-B.

- We develop a security- and network-performance-aware resource allocation strategy for IaaS cloud datacenters. It aims at improving network security and performance predictability offered to tenant applications by grouping them into virtual infrastructures.
- We formally present the proposed strategy as a Mixed-Integer Programming (MIP) optimization model and introduce a heuristic to efficiently allocate tenant applications in large-scale cloud platforms. Our strategy can be applied on different datacenter network topologies, such as today's multi-rooted trees [10] and richer topologies (e.g., VL2 [13] and Fat-Tree [14]).
- We evaluate the trade-off between the gain in security and performance for tenants and the cost imposed on cloud providers by our solution. Evaluation results show that the proposed approach can substantially increase security and performance with low extra cost.

The remainder of this paper is organized as follows. Section II discusses related work. Section III defines the threat model considered, while Section IV defines basic formulations related to the problem and which are later used throughout the paper. Section V presents our resource allocation strategy, and Section VI introduces a heuristic to efficiently allocate tenant applications. The evaluation of the proposed strategy appears in Section VII, and Section VIII closes the paper with final remarks and perspectives for future work.

II. Related Work

Providers use VLANs [15] in an attempt to isolate tenants (or applications) in the network. However, VLANs are not well-suited for cloud datacenter networks for two reasons. First, they use the Spanning Tree Protocol (STP), which cannot utilize the high capacity available in datacenter network topologies with rich connectivities (e.g., VL2 [13] and Fat-Tree [14]) [16]. Second, VLANs do not provide bandwidth guarantees.

Some recent work [8], [9] exploit the shared nature of the intra-cloud network to show how selfish and malicious tenants can perform several kinds of attacks, including DoS in the network. These attacks are made easier by the fact that providers do not charge for network traffic inside the cloud [3], [4].

HomeAlone [17], from another perspective, explores vulnerabilities of physical co-residence of VMs to increase the security of computational resources, which may end up improving application network performance as well. Therefore, HomeAlone is mostly orthogonal to our paper; in fact, it can be used together with our approach to improve application security and performance in cloud platforms.

Current resource allocation algorithms employed by cloud providers do not handle network security [18]. Such algorithms use round-robin across servers or across racks, taking into account only computational resources (processing power, memory and storage).

In this sense, the design of security- and network-performance-aware resource allocation strategies for cloud computing is a major research challenge. Recent work focuses on providing fair network sharing in accordance with weights assigned to VMs (or tenants) [1], [11] or on VM placement techniques [12], [19], [20]. These schemes, however, cannot protect the network from malicious insiders and may not prevent selfish behavior by applications running in the cloud platform. In particular, [1], [11] require substantial management overhead to control each VM network share, wasting resources.

SecondNet [4], Oktopus [10] and CloudNaaS [21], in turn, propose to allocate each application (or tenant) in a distinct virtual network (which is typically implemented by rate limiting techniques or by the Software Defined Networking approach). Nonetheless, these approaches present three drawbacks: i) low utilization of network resources, since each virtual network reserves bandwidth equivalent to the peak demand, yet most applications generate varying amounts of traffic in different phases of their execution [3], [7]; ii) high resource management overhead; and iii) (internal) fragmentation of both computational and network resources upon high rate of tenant arrival and departure (i.e., churn) [1]. These drawbacks hurt provider revenue and, ultimately, translate to higher tenant costs.

In general, cloud network resource sharing presents two shortcomings: i) network resources are scarce and often represent the bottleneck when compared to computational resources in datacenters [3], [13]; and ii) the lack of network isolation provide means for malicious parties to launch attacks against well-behaved tenants. Hence, our strategy addresses these issues in two ways. First, it minimizes the amount of bandwidth used by communication among VMs from the same application, thus saving network resources. Second, it is aware of both types of resources and isolates network ones among different application groups. This way, it may prevent attacks from untrusted tenants (or make these attacks ineffective), specially performance interference and denial of service threats.

III. Threat Model

We consider an IaaS tenant that operates one or more applications[2]. Each tenant has the same privileges as the others, similarly to [8]. In our model, adversaries are selfish and malicious parties. Selfish tenants launch performance interference attacks against other applications, increasing the network throughput of their VMs [22]. Malicious parties, in turn, cast several kinds of attacks on previously defined targets, including the extraction of confidential information from victims, man-in-the-middle, and DoS. To increase the effectiveness of an attack, malicious adversaries make use of placement techniques [8] to collocate their VMs near to the target.

Attacks against the availability of network resources are performed in two ways: i) by increasing the number of flows in the network, exploiting the lack of traffic isolation among applications [3]; and ii) by sending large UDP flows. Since all tenants share the same network (they compete for bandwidth

[2]Basically, each application consists of a collection of VMs. We will define an application in Section IV-A.

in the intra-cloud network), such attacks are not limited to their targets, but rather can affect a large number of applications.

The attacks considered can only be launched by an insider, that is, a tenant registered with the cloud. This requirement reduces, in theory, the likelihood of an attack, since the user should be accountable. However, in some platforms, accounts can be easily obtained (no strict requirements for accounts), or to compromise (user behavior) [23]. Furthermore, the detection of such attacks is not simple because of two reasons. First, an attacker can conceal malicious traffic as well-behaved [1]. Second, a persistent attacker is not easily deterred by obfuscation schemes [1] (i.e., techniques used by providers to hide resource location, for instance against network-based VM co-residence checks and internal cloud infrastructure mapping [8]).

IV. System Model

In this section, we define fundamental formulations used to model our strategy. The notations are represented by the following rules: i) superscripts s, v and r denote entries related to the physical substrate of the cloud platform, the virtual infrastructures and the requests from tenants, respectively; ii) subscripts are indices from attributes, variables or elements of a set. The notation is similar to that employed in [24].

A. Application Requests

The set of applications is denoted by A^r. An application request $a \in A^r$ from a tenant is defined by $< M_a^r,\ Band_a^r >$. The number of virtual machines is represented by M_a^r. Without loss of generality, we assume an homogeneous set of VMs, i.e., equal in terms of CPU, memory and storage consumption.

Apart from specifying the number of VMs, a request is extended to express network requirements. We provide tenants with a simple abstract view of the virtual network topology in which they reside. All VMs from the same application are represented as being connected to a virtual switch by a bidirectional link of capacity $Band_a^r \in \mathbb{R}^+$, as shown in Figure 1. This abstraction is motivated by the observation that, in private environments, tenants typically run their applications on dedicated clusters, with computational nodes connected through Ethernet switches [10].

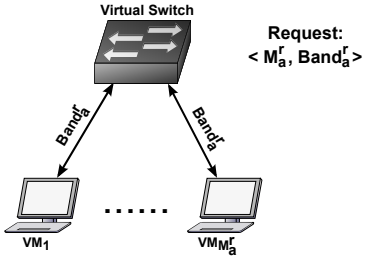


Fig. 1: Tenant's view of an application's network topology.

Trust relationships between applications are represented by $T^r_{a_i,a_j}$, which denotes whether application a_i from one tenant trusts application a_j from another tenant. We assume that trust relationships are direct, binary and symmetric. In other words, a tenant may or may not trust another tenant, with whom he interacts. If there is trust, then it is reciprocal.

These relationships can be established in two ways. First, they can be created based on the web of trust concept, similarly to a PGP-like scheme [25]. Second, the creation of trust relationships can be materialized by matching properties contained within SLAs signed by different customers and providers. This process would be assisted by the front-end responsible for receiving the requests and transferring them to the allocation module.

B. Virtual Infrastructures

A Virtual Infrastructure is composed of a set of virtual machines interconnected by a virtual network (virtual network devices and virtual links). It is a logically isolated domain with arbitrary topology (i.e., independent of the underlying cloud substrate). We model the set of VIs by I^v, where each VI $i \in I^v$ is a weighted bidirectional graph $G_i^v = < S_i^v, M_i^v, E_i^v, Band^v, Oversub^v >$. Without loss of generality, we assume the set of network devices (S_i^v) in i as switches, similarly to [10] and [3]. The subset of Top-of-Rack switches (ToR), in turn, is represented by $R_i^v \subset S_i^v$. The set of virtual machines in i is indicated by M_i^v, and the set of virtual links by E_i^v. Each link $e^v = (u, w) \in E_i^v$ connects nodes u and w ($u, w \in S_i^v \cup M_i^v$). Moreover, each link $e^v \in E_i^v$ has a bidirectional bandwidth $Band^v(e^v) \in \mathbb{R}^+$ and an oversubscription factor $Oversub^v(e^v) \in \mathbb{Z}^+$ employed by the provider to increase network resource utilization.

C. Physical Infrastructure

The physical substrate is composed of servers, network devices and links, similarly to [4]. This infrastructure is represented as a weighted bidirectional graph $G^s =< S^s, M^s, E^s, Band^s, Slots^s, Cost^s, Cap^s >$, where S^s is the set of network devices (switches), M^s is the set of servers, and E^s is the set of links. Each server $m^s \in M^s$ has $Slots^s(m^s) \in \mathbb{Z}^+$ slots. Each switch $s^s \in S^s$ has an associated number of virtual switches it can host ($Cap^s(s^s) \in \mathbb{Z}^+$), and a cost ($Cost^s(s^s) \in \mathbb{R}^+$) per virtual switch. The cost is proportional to the importance of the physical switch in the network (i.e., switches closer to the network core have higher utilization costs). The subset of ToR switches is represented by $R^s \subset S^s$. Each link $e^s = (u, w) \in E^s \mid u, w \in S^s \cup M^s$ between nodes u and w is associated with a bidirectional bandwidth $Band^s(e^s) \in \mathbb{R}^+$. Finally, $\mathcal{P}^s$ and $\mathcal{P}^s(u, w)$ denote, respectively, the set of all substrate paths and the set of substrate paths from source node u to destination node w.

V. Resource Allocation Strategy

In this section, we formally present our approach to allocate resources for incoming application requests at cloud platforms, which takes security and network-performance into account. Our strategy aims at mitigating the impact of attacks performed within the intra-cloud network. This is achieved by grouping applications into logically isolated domains (VIs) according to trust relationships between pairs of tenants and traffic generated between VMs of the same application.

To provide security- and network-performance-aware resource allocation, there is a fundamental challenge to be addressed: resource allocation with bandwidth-constrained network links is NP-Hard [16].

For this reason, we solve the problem by breaking it in two smaller steps (sub-problems), propose an allocation strategy for each one of them and, lastly, combine their results. An abstract view of the allocation process is shown in Figure 2, which contains three functions and a given set of virtual infrastructures. Function $\mathcal{H} : A^r \rightarrow G^s$ is the approach employed by current public cloud providers, which maps applications directly into the physical substrate (considering only computational resources). Unlike this approach, we consider the network in the allocation process and decompose $\mathcal{H}$ in $\mathcal{F}$ and $\mathcal{G}$, as follows. Function $\mathcal{F} : A^r \rightarrow I^v$ distributes and maps applications into virtual infrastructures. Function $\mathcal{G} : I^v \rightarrow G^s$ allocates each virtual infrastructure onto the physical substrate. Note that, in theory, $\mathcal{F}$ and $\mathcal{G}$ can be executed in arbitrary order, but in practice $\mathcal{G}$ can be used first to allocate the VIs on the cloud substrate whereas $\mathcal{F}$ can be used later to allocate every incoming application request.

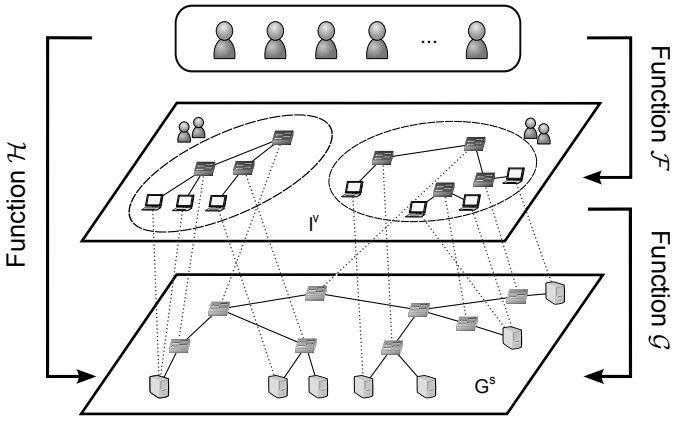


Fig. 2: Cloud resource allocation overview.

In this paper, we focus on the problem of solving both $\mathcal{F}$ and $\mathcal{G}$ for a given set of virtual infrastructures. In doing so, we take as input the set of virtual infrastructures (more specifically, how many and of which size each). This could be determined arbitrarily by the provider, or for example based on the resource utilization history [19], with information from already allocated applications collected by tools such as Amazon CloudWatch[3]. The next two subsections describe, respectively, functions $\mathcal{G}$ and $\mathcal{F}$.

A. Mapping VIs onto the Physical Substrate

The mapping of virtual infrastructures on the cloud substrate (virtual to physical mapping) is performed by Function $\mathcal{G}$. It addresses a problem similar to Virtual Network Embedding (VNE) [24], which allows the strategy to deal with the heterogeneity of virtual topologies in comparison to the physical substrate topology. Unlike VNE, the allocation also takes computational nodes (physical and virtual machines) into account.

Input. This function receives as inputs the set of virtual infrastructures (Section IV-B) and the physical substrate (Section IV-C). Furthermore, parameter $\alpha_{(w_1,w_2)}$ quantifies the importance of link (w_1, w_2) within the $(0, 1]$ range.

Variables. The main variables used by Function $\mathcal{G}$ are:

- $x_{i,s^v,s^s} \in \mathbb{B}$: indicates if virtual switch $s^v \in S_i^v$ from virtual infrastructure $i \in I^v$ is allocated at physical switch $s^s \in S^s$;
- $y_{i,e^v,(w_1,w_2)} \in \mathbb{B}$: indicates if virtual link $e^v \in E_i^v$, which belongs to virtual infrastructure $i \in I^v$, uses physical link $(w_1, w_2) \in E^s$.

Objective. Equation (1) minimizes the amount of physical resources used to allocate the virtual infrastructures. That is, we seek to minimize the total amount of bandwidth consumed from the substrate. By dividing $\alpha_{(w_1,w_2)}$ by the total capacity of link (w_1, w_2), we ensure that links with lower importance and greater capacity are preferred.

$$Z = \text{Min} \sum_{(w_1,w_2)\in E^s} \frac{\alpha_{(w_1,w_2)}}{Band^s(w_1,w_2)} * \sum_{i\in I^v} \sum_{e^v \in E_i^v} y_{i,e^v,(w_1,w_2)} * Band^v(e^v) \quad (1)$$

The set of constraints guide the allocation process. The assignment of each VI to the substrate can be decomposed in two major components, as follows.

Node assignment. Each virtual node (virtual switch or VM) is assigned to a substrate node by mapping $\mathcal{M}_n : (M_i^v \cup S_i^v) \rightarrow (M^s \cup S^s),\ \forall i \in I^v$ from virtual nodes to substrate nodes:

$$\mathcal{M}_n(m^v) \in M^s \mid m^v \in M_i^v \text{ or}$$
$$\mathcal{M}_n(r^v) \in R^s \mid r^v \in R_i^v \text{ or}$$
$$\mathcal{M}_n(s^v) \in S^s \mid s^v \in S_i^v \backslash R_i^v$$

Link assignment. Each virtual link is mapped to a single substrate path (unsplittable flow) between the corresponding substrate nodes that host the virtual nodes at the ends of the virtual link. The assignment is defined by mapping $\mathcal{M}_e : E_i^v \rightarrow \mathcal{P}^s,\ \forall i \in I^v$ from virtual links to substrate paths such that for all $e^v = (w_1, w_2) \in E_i^v,\ \forall i \in I^v$:

$$\mathcal{M}_e(w_1, w_2) \subseteq \mathcal{P}^s\left(\mathcal{M}_n(w_1), \mathcal{M}_n(w_2)\right)$$

subject to $Resid(p^s) \geq Band^v(e^v) \mid p \in \mathcal{M}_e(e^v)$, where $Resid(p^s)$ denotes the residual (spare) capacity of substrate path p^s.

The constraints from Function $\mathcal{G}$ ensure the correct mapping of virtual resources to the physical substrate, but are omitted due to space limits.

B. Mapping Applications into Virtual Infrastructures

Function $\mathcal{F}$ maps applications into VIs according to the mutual trust among tenants and the bandwidth consumed by communication among VMs of the same application.

Input. This function receives as input an incoming application request (Section IV-A), the set of virtual infrastructures (Section IV-B), two parameters (γ and δ) and sets $\mathcal{P}_i^v,\ \forall i \in I^v$, as follows. γ and δ are used to balance both components of the optimization objective. Set $\mathcal{P}_i^v$ consists of all pairs of racks

[3]http://aws.amazon.com/cloudwatch/

from VI $i \in I^v$ [$p = (r_1, r_2) \in \mathcal{P}_i^v \mid r_1, r_2 \in R_i^v$ and $r_1 \neq r_2$] and is used to calculate the maximum bandwidth necessary for communication between VMs of the same application allocated at distinct racks.

Variables. The main variables are:

- $H_{i,a_1,a_2} \in \mathbb{B}$: indicates whether applications a_1 and a_2 are allocated at VI $i \in I^v$;
- $F_{a,i,(w_1,w_2),p} \in \mathbb{R}^+$: indicates the total amount of bandwidth required by application a at link $(w_1, w_2) \in E_i^v \mid w_1, w_2 \in S_i^v$ for communication between its VMs allocated at racks $r_1, r_2 \in R_i^v \mid p = (r_1, r_2) \in \mathcal{P}_i^v$ from VI $i \in I^v$. The amount of bandwidth is defined according to the number of VMs of application a at r_1 and r_2 and will be explained later [Equation (3)].

Objective. Equation (2) addresses two keys properties of cloud computing: security and performance. Security is increased by minimizing the number of mutually untrusted relationships inside each VI (i.e., maximizing mutual trust among applications inside VIs). Performance, in turn, is increased because of two reasons. First, we cluster VMs from the same application, reducing the amount of network resources needed by communication between these VMs. Second, we isolate mutually untrusted tenant applications in distinct VIs. Thereby, applications are less susceptible to attacks in the network, specially performance interference and DoS.

$$Z = \text{Min } \gamma * \left(\sum_{i \in I^v} \sum_{a_1 \in A^r} \sum_{a_2 \in A^r} (1 - T^r_{a_1,a_2}) * H_{i,a_1,a_2} \right) + \delta \left(\sum_{a \in A^r} \sum_{i \in I^v} \sum_{(w_1,w_2) \in E_i^v} \sum_{p \in \mathcal{P}_i^v} F_{a,i,w_1,w_2,p} \right) \quad (2)$$

Next, we discuss two aspects of our model: inter-rack bandwidth consumption and path selection.

Inter-rack bandwidth consumption. The cost of communication between VMs positioned in the same rack is negligible, since traffic remains internal to the rack and uses only links that connect those VMs to the ToR switch. In contrast, traffic between VMs from different racks imposes a cost, which is given by the bandwidth consumed and the set of links used.

We minimize the latter cost by employing the concept of VM clusters[4]. A VM cluster consists of a set of VMs of the same application located in the same rack. Therefore, we aim at allocating each application into few, close VM clusters to avoid spending extra bandwidth for communication. This also benefits future allocations by saving network resources.

When all VMs of the same application are placed into the same VM cluster, all traffic is kept within the ToR (i.e., negligible cost). However, if the set of VMs is distributed into more than one VM cluster, we must ensure that there is enough available bandwidth for communication between these clusters. For instance, consider the scenario presented in Figure 3, where two applications have two VM clusters each:

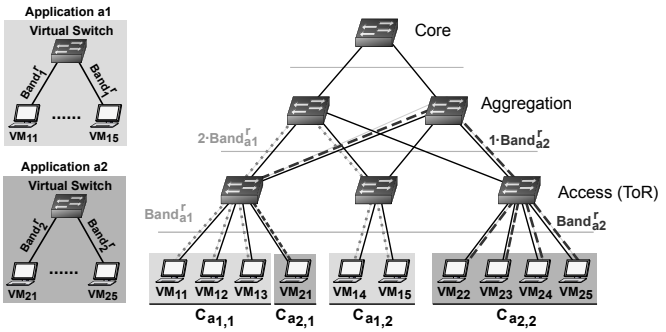


Fig. 3: Communication between VM clusters.

we must guarantee network bandwidth in all links of a path connecting the pairs of VM clusters from each application. Since a single VM of application a_1 cannot send or receive data at a rate greater than $Band^r_{a_1}$, traffic between the pair of clusters $C_{a_1,1}$ and $C_{a_1,2}$ is limited by the cluster with the lowest rate: $\min(|C_{a_1,1}|, |C_{a_1,2}|) * Band^r_{a_1}$. Thus, the bandwidth required by one VM cluster to communicate with all other clusters of the same application is given by the following expression:

$$B_{a_x,c_i} = \min \left(|c_i| * Band^r_{a_x}, \sum_{c \in C^r_{a_x}, c \neq c_i} |c| * Band^r_{a_x} \right) \quad \forall c_i \in C^r_{a_x} \quad (3)$$

where B_{a_x,c_i} denotes the bandwidth required by the i^{th} VM cluster to communicate with other clusters from application a_x.

Path selection. The model presented in this paper considers the use of one path for communication between pairs of VM clusters from the same application. That is, datacenters typically have networks with rich connectivity, such as multi-rooted trees [26] and Fat-Tree [14]. Cloud network devices are usually connected with several links that are balanced by multipathing techniques, such as Equal-Cost Multi-Path (ECMP) [7] and Valiant Load Balancing (VLB) [3]. Nevertheless, given the amount of multiplexing over the links and the limited number of paths, these multiple paths can be considered as a single aggregate link for bandwidth reservation [10].

VI. Embedding Heuristic

Based on the formal model presented in Section V, in this section we introduce a constructive heuristic for Function $\mathcal{F}$ to efficiently allocate tenant applications in cloud platforms. Note that we do not present a heuristic for Function $\mathcal{G}$ for two reasons. First, Function $\mathcal{G}$ is executed only when VIs are allocated on the cloud substrate. Thus, we can use a solver, such as CPLEX[5], to optimally perform this operation. Second, should a heuristic be necessary, there are several virtual network embedding algorithms in the state-of-the-art,

[4]This concept is similar to that of VM grouping, used in [10]. We prefer the term *VM cluster* so to avoid confusion with application grouping.

[5]http://www-01.ibm.com/software/integration/optimization/cplexoptimization-studio/

for instance [24]. These algorithms can be easily adapted to embed VIs in cloud infrastructures.

Function $\mathcal{F}$, in turn, may take a considerable amount of time to allocate one application in the cloud when executed by a solver, specially in large-scale cloud datacenters. However, the high rate of tenant arrival and departure requires the operation to be performed as quickly as possible. Hence, we design a constructive heuristic, which is shown in Algorithm 1.

The key ideia is based on two factors. First, we quantify the number of mutually untrusted relationships inside each VI for the incoming request (we seek to increase security by avoiding mutually untrusted tenants to be allocated in the same VI). Second, in the VI with the lowest number (that is, the highest security), we allocate the application VMs as close as possible to each other and in the smallest number of VM clusters. Thus, we can increase security and, at the same time, minimize bandwidth consumption for intra-application communication.

Algorithm 1: Application allocation algorithm.

Input : Application a, Virtual infrastructure set I^v

```
1  unvisitedVIs ← GetVIs(I^v);
2  while true do
3      i ← SelectVI(unvisitedVIs, T^r_{a,x});
4      if not i then return false;
5      unvisitedVIs ← unvisitedVIs \ {i};
6      C^r_a ← ∅;
7      r^v ← FindBestRack(i, M^r_a);
8      maxVMs ← MaxAvailableCluster(r^v, M^r_a, Band^r_a);
9      C^r_a ← C^r_a ∪ {Cluster(r^v, maxVMs)};
10     allocatedVMs ← maxVMs;
11     if allocatedVMs < M^r_a then
12         switchQueue ← FindNeighborSwitches(r^v);
13         while allocatedVMs < M^r_a do
14             s^v ← GetSwitch(switchQueue);
15             if not s^v then break;
16             switchQueue ← FindNeighborSwitches(s^v);
17             torQueue ← FindRacks(s^v);
18             while torQueue not empty do
19                 r^v ← GetToR(torQueue);
20                 maxVMs ← MaxAvailableCluster(r^v,
                   (M^r_a − allocatedVMs), Band^r_a);
21                 C^r_a ← C^r_a ∪ Cluster(r^v, maxVMs);
22                 allocatedVMs ← allocatedVMs + maxVMs;
23                 if allocatedVMs == M^r_a then break;
24             end while
25         end while
26     end if
27     if allocatedVMs == M^r_a and AllocBandwidth(C^r_a) then
28         return true;
29     end if
30 end while
```

The algorithm works as follows. First, it creates a list (`unvisitedVIs`) of all VIs with enough available VMs to hold the request (line 1). Then, it verifies one VI at a time from the list in an attempt to allocate the incoming application (lines 2 – 30). To this end, function `SelectVI` selects one VI based on two factors: i) the number of mutually untrusted relationships; and ii) the number of available VMs (line 3). It selects the VI with the lowest number of mutually untrusted relationships between the incoming request and the tenant applications already allocated in the VI. If there are more than one VI with the lowest number, it will choose the one with the largest number of available VMs. In doing so, we take security into account while augmenting the possibility of allocating all VMs from the application close to each other in order to address network performance as well.

The algorithm, then, initializes the set of VM clusters C^r_a for the application (line 6) and calls function `FindBestRack` (line 7). This function selects one rack in the following way: if the number of unallocated VMs is smaller than the number of VMs per rack, it tries to find a rack with the closest number of available VMs (which must be enough to allocate the entire application), in order to create a single cluster; otherwise, it employs a greedy behavior, that is, it selects one of the racks with the largest number of available VMs. When a rack is chosen, function `MaxAvailableCluster` verifies the maximum cluster size that the rack can hold (line 8) and the cluster is created (line 9).

Next, if there are still unallocated VMs, the algorithm will search for racks close to the already allocated VM cluster (lines 13 – 25). This step is performed by verifying directly connected switches (function `FindNeighborSwitches`) from r^v and racks connected to the topology lower levels of these switches (function `FindRacks`). When a rack with available capacity is found, a new VM cluster is created for the application. This process is repeated until all requested VMs are allocated.

Finally, after all VMs have been mapped inside the VI, function `AllocBandwidth` calculates and allocates the bandwidth necessary for communication among VM clusters from the incoming application (line 27). Upon successfully allocating the bandwidth required by communication among the clusters, the algorithm returns a success code. In contrast, if the selected VI was not able to hold the request due to the lack of available resources, the algorithm attempts to allocate the incoming application to another VI. In case that all VIs were verified and none of them had enough residual resources to allocate the request, the operation fails and the request is discarded.

VII. Evaluation

In this section, we first describe the evaluation environment and then present the main results. The evaluation of our approach focuses primarily on quantifying the trade-off between the gain in security and performance to tenants and the cost (internal resource fragmentation) it imposes on cloud providers.

A. Evaluation Setup

To show the benefits of our approach in large-scale cloud platforms, we developed a simulator that models a multi-tenant shared datacenter. We focus on tree-like topologies such as multi-rooted trees used in today's datacenters [1]. The network topology consists of a three-level tree topology, with 16,000 machines at level 0, each with 4 VM slots (i.e., with a total amount of 64,000 available VMs in the cloud platform). Each rack is composed of 40 machines linked to a ToR switch. Every 20 ToR switches are connected to an aggregation switch,

which, in turn, is connected to the datacenter core switches. This setup is similar to [3], [10].

Workload. The workload is composed by requests of applications to be allocated in the cloud platform. Unless otherwise specified, it is defined as follows. The number of VMs and bandwidth of each request is exponentially distributed around a mean of $\lambda = 49$ VMs (which is consistent with what is observed in cloud datacenters [1]) and uniformly distributed in the interval $[1, 500]$ Mbps, respectively. Mutual trust between tenants was generated through direct relationships between them in a random graph with degree of each vertex (tenant) following a distribution $P(k) \propto \frac{1}{k}$. Each virtual infrastructure from the set I^v, in turn, is defined as a tree-like topology with similar size in comparison to the other VIs from the set.

B. Evaluation Results

Improved security and performance for tenants. Security is quantified by measuring the number of mutually untrusted tenants assigned to the same VI. It is desirable to have this value minimized, because it may expose applications to several kinds of attacks, including performance interference ones caused by untrusted tenants. We verify trust relationships between tenants in two scenarios: when allocating batches of applications (that is, when all application requests are known beforehand, in an offline setting), and when applications arrive without prior knowledge (i.e., in an online setting). We further show how performance interference attacks are reduced. Our results are compared to the baseline scenario (current cloud allocation scheme) in which all tenants share the same network.

Figure 4 depicts the variation of mutually untrusted relationships for three batches of application requests in accordance with the number of VIs offered by the provider. Results show that the number of applications is not the main factor to increase security, but rather the number of VIs offered by the provider. In general, we find that the number of mutually untrusted relationships decreases, and thereby security increases, with a logarithmic behavior according to the number of VIs.

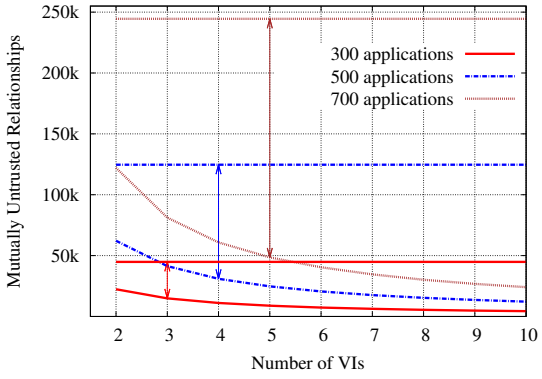


Fig. 4: Security when allocating batches of applications.

Next, we measure how security increases when application requests arrive without prior knowledge. The arrival rate of each request is given by a Poisson distribution (similarly to [3] and [10]) with an average of 10 requests per time unit. In this scenario, we adopt a common admission control (similar to that of Amazon EC2), which rejects an application request that cannot be allocated upon its arrival.

Figure 5 shows how the number mutually untrusted relationships inside the cloud varies over 2,000 time units. The number of mutually untrusted relationships (Y-axis) is represented in logarithmic scale, as these numbers differ significantly for different sets of VIs. At first, the number of mutually untrusted relationships increases because all incoming applications are allocated, since there are ample resources. As time passes and the cloud-load increases (less available resources), this number tends to stabilize, because new applications are allocated only when already allocated applications conclude their execution and are deallocated (which releases resources). We find that the higher the isolation among tenant applications, the greater the security, since the number of mutually trusting applications inside each VI is maximized and, thus, performance interference attacks are minimized. However, the level of security offered by the provider tends to stabilize after a certain number of VIs, because security increases with a logarithmic behavior.

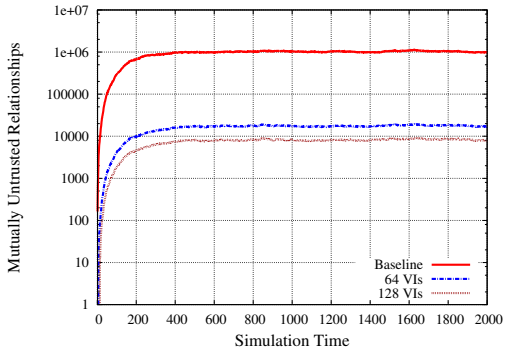


Fig. 5: Security in an online setting.

We also verify the number of applications competing for bandwidth in level-2 and level-3 links of the virtual tree topologies. Figure 6 depicts the mean number of application sharing level-2 links (i.e., links between ToR and aggregation switches) over 2,000 time units, while Figure 7 shows the mean number of applications sharing level-3 links (i.e., links between aggregation and core switches). Level-3 links are shared by a larger number of applications than level-2 links because layer-3 switches interconnect several layer-2 switches and, as time passes, the arrival and departure of applications lead to dispersion of available resources in the infrastructure; thus each incoming application may be allocated in several racks from different aggregation switches (and VMs from distinct racks communicate with other VMs of the same application through level-3 links). We see that the use of VIs can greatly reduce the number of applications competing for bandwidth in level-2 and, in particular, in level-3 links. This reduction greatly minimizes performance interference attacks. Thereby, it can increase overall application performance by improving application network performance, since performance interference in the network is one of the leading

causes for poor application performance in the cloud [1], [10]. We achieve this by completely isolating VIs from one another, that is, there is no competition for network resources among VIs, but rather only inside VIs.

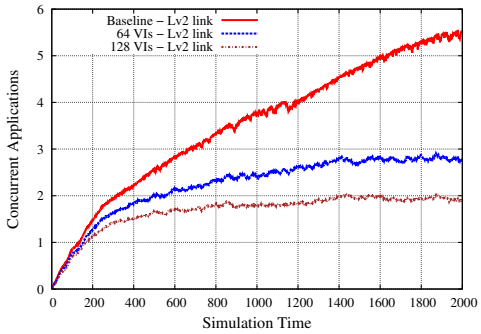


Fig. 6: Mean number of applications sharing a level-2 link.

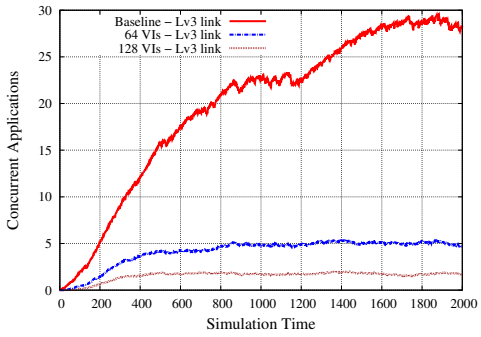


Fig. 7: Mean number of applications sharing a level-3 link.

Resource fragmentation. The creation of VIs and grouping of tenant applications may restrain the allocation of requests, because of (internal) resource fragmentation. Specifically, fragmentation happens when the sum of available resources (considering all VIs) would be enough to accept the incoming request, but no VI alone has the amount of resources available to accept the request.

Figure 8 shows results regarding internal fragmentation of resources. Figure 8(a) shows the overall acceptance ratio of application VM requests according to the number of VIs. We present results for applications with the number of VMs (λ) exponentially distributed around different means. We verify that the acceptance ratio decreases linearly according to the number of VIs. For requests with exponential mean of $\lambda = 29$, there exists negligible fragmentation, since the acceptance ratio does not decrease with 128 VIs in comparison to the baseline scenario. In contrast, there is some fragmentation when the number of VMs is distributed around higher λ, since there is a reduction in the acceptance ratio (2.92% with $\lambda = 89$, 4.26% with $\lambda = 69$ and 6.06% with $\lambda = 49$) when comparing the baseline with 128 VIs. Thus, the excessive use of virtual infrastructures may lead to resource fragmentation inside the cloud infrastructure. However, it is small even for a worst-case scenario with 128 VIs.

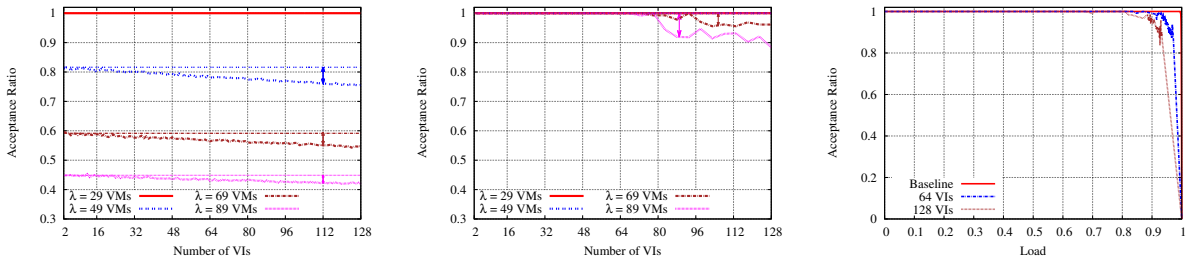
Figure 8(b) depicts the acceptance ratio with cloud-load between 70% and 80% (i.e., the usual load of public cloud platforms, such as Amazon EC2 [27]). We see that the acceptance ratio decreases according to the number of VIs offered and the size of applications (that is, the bigger the applications allocated, the worse the fragmentation). Although the fragmentation tends to increase significantly with the number of VMs distributed around $\lambda = 89$ (but still with high acceptance ratio), note that this is a worst-case value.

Figure 8(c), in turn, shows the acceptance ratio of requests for $\lambda = 49$ (i.e., a setting that is observed is cloud datacenters [1]) according to the cloud-load for different sets of VIs. Notice that the acceptance ratio drops significantly after the cloud-load goes over 97% for 64 VIs (92% for 128 VIs). Since providers usually operate their cloud datacenters at 70-80% occupancy [27], we consider that this burden is pragmatically negligible. Furthermore, our strategy minimizes resource fragmentation by assigning VMs from the same application to VM clusters, thus reducing the amount of network resources consumed for intra-application communication and saving network resources for future allocations. Overall, it is possible to substantially increase security with minimum addition of resource fragmentation in comparison to the baseline scenario. Providers do not need to offer a huge number of VIs, because security increases with logarithmic behavior. The trade-off between security and cost, if well explored, can lead to an attractive configuration between the number of VIs offered (security and performance) and resource fragmentation (cost).

Provider Revenue. Cloud providers, such as Amazon EC2, charge tenants solely based on the time they occupy their VMs. However, we envision that, in the future, cloud providers will charge for VM-time *and* network bandwidth. Since it is still ongoing research [28], we adopt a simple pricing model similar to [3], [10], which effectively charges both computation and networking. Hence, a tenant using M_a^r VMs for time T pays $M_a^r \times T(k_v + k_b \times Band_a^r)$, where k_v is the unit-time VM cost and k_b is the unit-volume bandwidth cost. Such pricing model can be used as long as the provider handles network resource allocation. This way, we compare provider revenue for the baseline scenario under today's charging model against our approach under both pricing models (with and without considering bandwidth). Figure 9 shows the revenue of our approach as a percentage of the baseline. We see that provider revenue decreases around 3.5% with 64 VIs (6% with 128 VIs) in comparison with today's charging model. Nonetheless, it can be substantially increased (about 17.5% for both 64 and 128 VIs) with a networking-aware pricing model.

VIII. Conclusions and Future Work

In this paper, we propose a resource allocation strategy that increases the security of cloud network resource sharing and application performance. Security is increased by isolating applications from mutually untrusted tenants, which reduces the impact of selfish and malicious behavior in the network. Application performance is augmented by clustering VMs from the same application and by minimizing performance interference attacks from untrusted tenants. Thus, the environment becomes more resilient against tenants that could hurt



(a) Overall acceptance ratio according to the number of VIs.

(b) Acceptance ratio according to the number of VIs for cloud-load between 70-80%.

(c) Acceptance ratio according to the cloud-load ($\lambda = 49$ VMs).

Fig. 8: Internal resource fragmentation.

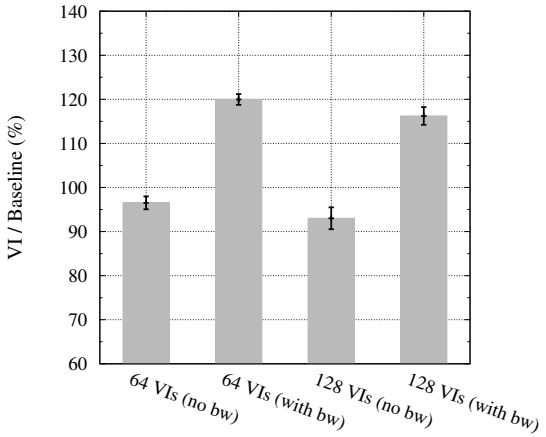


Fig. 9: Provider revenue.

other applications. We evaluated and compared our strategy with a baseline scenario, in which all applications share the same network. Our results show that security and performance are improved with little extra cost for the provider.

In future work, we intend to dynamically reduce or increase virtual infrastructure size in order to improve security and minimize resource fragmentation. We further aim at including security requirements when mapping VIs onto the physical substrate (i.e., function $\mathcal{G}$), thus improving isolation among VIs, and exploring virtual link embedding into multiple paths.

References

[1] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, "Sharing the data center network," in *USENIX NSDI*, 2011.

[2] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "Faircloud: sharing the network in cloud computing," in *ACM SIGCOMM*, 2012.

[3] D. Xie, N. Ding, Y. C. Hu, and R. Kompella, "The only constant is change: Incorporating time-varying network reservations in data centers," in *ACM SIGCOMM*, 2012.

[4] 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 *ACM CoNEXT*, 2010.

[5] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," 2008.

[6] E. Kohler, M. Handley, and S. Floyd, "Datagram Congestion Control Protocol (DCCP)," 2006.

[7] D. Abts and B. Felderman, "A guided tour of data-center networking," *Comm. ACM*, vol. 55, no. 6, Jun. 2012.

[8] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *ACM CCS*, 2009.

[9] H. Liu, "A new form of DOS attack in a cloud and its avoidance mechanism," in *ACM CCSW*, 2010.

[10] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *ACM SIGCOMM*, 2011.

[11] V. T. Lam, S. Radhakrishnan, R. Pan, A. Vahdat, and G. Varghese, "Netshare and stochastic netshare: predictable bandwidth allocation for data centers," *ACM SIGCOMM CCR*, vol. 42, no. 3.

[12] D. Breitgand and A. Epstein, "Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds," in *IEEE INFOCOM*, 2012.

[13] 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," in *ACM SIGCOMM*, 2009.

[14] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM*, 2008.

[15] "IEEE std. 802.1Q, Virtual Bridged Local Area Networks," IEEE Computer Society, 2005.

[16] 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," Microsoft Research, Technical Report MSR-TR-2010-81, 2010.

[17] Y. Zhang, A. Juels, A. Oprea, and M. Reiter, "Homealone: Co-residency detection in the cloud via side-channel analysis," in *IEEE Symposium on Security and Privacy (SP)*, 2011.

[18] I. Kitsos, A. Papaioannou, N. Tsikoudis, and K. Magoutis, "Adapting data-intensive workloads to generic allocation policies in cloud infrastructures," in *IEEE/IFIP NOMS*, 2012.

[19] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *IEEE INFOCOM*, 2010.

[20] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *IEEE INFOCOM*, 2011.

[21] T. Benson, A. Akella, A. Shaikh, and S. Sahu, "CloudNaaS: a cloud networking platform for enterprise applications," in *ACM SOCC*, 2011.

[22] M. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and F. Zhani, "Data center network virtualization: A survey," *IEEE Comm. Surv. Tut.*, vol. PP, no. 99, 2012.

[23] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM CCR*, vol. 39, no. 1, 2008.

[24] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *IEEE INFOCOM*, 2009.

[25] P. R. Zimmermann, *The official PGP user's guide*. Cambridge, MA, USA: MIT Press, 1995.

[26] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul, "Spain: Cots data-center ethernet for multipathing over arbitrary topologies," in *USENIX NSDI*, 2010.

[27] R. Bias, "Amazon's EC2 Generating 220M," Cloudscaling, 2011. [Online]. Available: http://goo.gl/d1Vai.

[28] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "The price is right: towards location-independent costs in datacenters," in *ACM HotNets*, 2011.