UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

DANIEL STEFANI MARCON

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

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Prof. Dr. Marinho Pilla Barcellos
Advisor

Porto Alegre, March 2013

CIP – CATALOGING-IN-PUBLICATION

Marcon, Daniel Stefani

Trust-based Application Grouping for Cloud Datacenters: improving security in shared infrastructures / Daniel Stefani Marcon. – Porto Alegre: PPGC da UFRGS, 2013.

96 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2013. Advisor: Marinho Pilla Barcellos.

1. Cloud computing. 2. Resource allocation. 3. Intra-cloud network sharing. 4. Security. 5. Performance interference. 6. Denial of service. I. Barcellos, Marinho Pilla. II. Título.

"Everything is possible for one who believes."
— Jesus (Mark 9:23)

ACKNOWLEDGEMENTS

First of all, I thank God for always being with me and the help He has given me throughout my entire life.

Thanks to my godmother Darcila Stefani and to my godfather Fausto Stefani for always supporting me and, especially, for all the help they provided me during my masters degree studies.

Thanks to my advisor Marinho Barcellos for all the guidance on how to conduct a high level research, on how to write scientific texts correctly and for teaching me how to develop critical thinking. I am also grateful to Profs. Luciano Gaspary and Luciana Buriol for all the contribution they gave to my thesis.

Finally, I thank all my friends who helped me during this work, directly or indirectly.

AGRADECIMENTOS

Agradeço, em primeiro lugar, a Deus, por estar sempre comigo e ter me proporcionado finalizar este trabalho.

Agradeço à minha dinda-mãe Darcila Stefani e ao meu dindo Fausto Stefani por terem me apoiado durante toda a minha vida e, em especial, pela ajuda em todos os momentos do curso de mestrado.

Agradeço ao meu orientador, Prof. Marinho Barcellos, por toda a orientação durante o mestrado: pelos ensinamentos sobre como realizar uma pesquisa de alto nível, como escrever textos científicos corretamente e como desenvolver o pensamento crítico. Também agradeço aos Profs. Luciano Gaspary e Luciana Buriol pelos conselhos, correções e sugestões sobre o trabalho.

Por fim, agradeço a todos os meus amigos que me ajudaram durante o período de mestrado, direta ou indiretamente.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

DCCP	Datagram Congestion Control Protocol
DC	Datacenter
DCN	Datacenter Network
DoS	Denial of Service
EC2	Elastic Compute Cloud
ECMP	Equal-Cost Multi-Path
GLPK	GNU Linear Programming Kit
IaaS	Infrastructure as a Service
MIP	Mixed-Integer Programming
RTT	Round Trip Time
SBP	Stochastic Bin Packing
STP	Spanning Tree Protocol
TCP	Transmission Control Protocol
TFRC	TCP-Friendly Rate Control
TIVC	Time-Interleaved Virtual Cluster
UDP	User Datagram Protocol
VC	Virtual Cluster
VDC	Virtual Datacenter
VOC	Virtual Oversubscribed Cluster
VI	Virtual Infrastructure
VLAN	Virtual Local Area Network
VLB	Valiant Load Balancing
VM	Virtual Machine

LIST OF FIGURES

LIST OF TABLES

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. Recent research has proposed resource allocation algorithms that cannot protect tenants against attacks in the network or result in underutilization of resources. In this thesis, we introduce a resource allocation strategy that increases the security of network resource sharing among tenant applications. This is achieved by grouping applications from mutually trusting users into logically isolated domains composed of a set of virtual machines as well as the virtual network interconnecting them (virtual infrastructures - VIs), while considering the amount of traffic generated by the communication between VMs from the same application. Due to the hardness of the cloud resource allocation problem, we decompose the strategy in two steps. The first one allocates a given set of VIs onto the physical substrate, while the second distributes and maps applications into the set of virtual infrastructures. The use of VIs provides some level of isolation and higher security. However, groups may lead to fragmentation and negatively affect resource utilization. Therefore, 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 additional cost. In particular, the security can be logarithmically increased according to the number of VIs, while internal resource fragmentation linearly grows as the number of VIs offered by the provider increases.

Keywords: Cloud computing, Resource allocation, Intra-cloud network sharing, Security, Performance interference, Denial of service.

Agrupamento de Aplicações Baseado em Relações de Confiança para Datacenters de Nuvens: Aumentando a Segurança em Infraestruturas Compartilhadas

RESUMO

A computação em nuvem é um paradigma que tem atraído uma grande quantidade de clientes por meio do oferecimento de recursos computacionais através de um modelo de pagamento pelo uso. Entretanto, o compartilhamento da rede interna da nuvem por todos os locatários possibilita que usuários utilizem de forma egoísta ou maliciosa os recursos da rede, ocasionando ataques contra a privacidade e a integridade dos dados e a disponibilidade dos recursos. Os algoritmos de alocação atuais não impedem que a disponibilidade dos recursos de rede seja afetada por ataques ou resultam em subutilização de recursos. Nessa dissertação, é proposta uma estratégia para a alocação de recursos que aumenta a segurança no compartilhamento da rede da nuvem entre as aplicações de locatários. Esse objetivo é alcançado por meio do agrupamento de aplicações provenientes de usuários mutuamente confiáveis em domínios logicamente isolados, compostos por um conjunto de máquinas virtuais interconectadas por uma rede virtual (infraestruturas virtuais – VIs), além de considerar-se a quantidade de tráfego gerada pela comunicação entre VMs da mesma aplicação. Devido à complexidade do problema de alocação de recursos em nuvens computacionais, a estratégia é decomposta em duas etapas. Na primeira, dado um conjunto pre-estabelecido de VIs, alocam-se as mesmas no substrato físico, enquanto a segunda distribui e mapeia as aplicações no conjunto de infraestruturas virtuais. O uso de VIs provê um maior nível de isolamento entre locatários e, consequentemente, maior segurança. Contudo, o agrupamento pode resultar em fragmentação e afetar negativamente o grau de utilização dos recursos. Dessa forma, estuda-se esse compromisso e a factibilidade da abordagem proposta. Os resultados mostram os benefícios da estratégia de alocação proposta, que oferece maior proteção aos recursos de rede com baixo custo extra. Em particular, a segurança aumenta logaritmicamente de acordo com o número de VIs, enquanto a fragmentação de recursos cresce linearmente de acordo com o aumento do número de VIs oferecidas pelo provedor.

Palavras-chave: Computação em nuvem, Alocação de recursos, Compartilhamento de recursos da rede interna da nuvem, Segurança, Interferência de desempenho, Negação de serviço.

1 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 increase resource utilization, reduce operational costs and, thus, achieve economies of scale by implementing cloud datacenters as highly multiplexed shared environments, with different applications coexisting on the same set of physical resources (ARMBRUST et al., 2009, 2010).

Providers, however, lack mechanisms to capture and control network requirements of the interactions among allocated virtual machines (VMs) (POPA et al., 2012; XIE et al., 2012; GUO et al., 2010a). For example, congestion control mechanisms used in intra-cloud networks, such as TCP-Friendly Rate Control (TFRC) (FLOYD et al., 2008) and Datagram Congestion Control Protocol (DCCP) (KOHLER; HANDLEY; FLOYD, 2006), do not ensure robust traffic isolation among different applications, especially with distinct bandwidth requirements (ABTS; FELDERMAN, 2012b). 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 (RISTENPART et al., 2009; SHIEH et al., 2011; LIU, 2010). Selfish attacks are characterized by the consumption of an unfair share of the network (i.e., performance interference attacks (SHIEH et al., 2011)), 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 (XIE et al., 2012). Providers, in turn, lose revenue, because attacks can affect network availability and reduce datacenter throughput (BALLANI et al., 2011b).

Vulnerabilities of cloud network resource sharing have been studied in Liu (2010), Ristenpart et al. (2009) and Shieh et al. (2011). Recent proposals try to increase network security through network-aware resource allocation strategies (SHIEH et al., 2011; LAM et al., 2012; BREITGAND; EPSTEIN, 2012; BALLANI et al., 2011b). Nonetheless, these schemes can neither protect the network from malicious insiders nor prevent selfish behavior by other applications running in the cloud environment

or result in underutilization of resources.

In this thesis, 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 the impact of 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 (seeking to reduce costs at the expense of other applications). 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 Ballani et al. (2011b) and Xie et al. (2012).

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 thesis are summarized as follows:

- 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 (BALLANI et al., 2011b) and richer topologies (e.g., VL2 (GREENBERG et al., 2009) and Fat-Tree (AL-FARES; LOUKISSAS; VAHDAT, 2008)).

- 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 (resource fragmentation).

The remainder of this thesis can be outlined as follows. Chapter 2 presents a review of the problem being addressed and summarizes related work. Chapter 3

[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 Chapter 3.

defines the basis in which our approach was developed, that is, the threat model considered and the basic formulations related to the problem, which are later used throughout the thesis. Chapter 4 formally presents our resource allocation strategy as a Mixed-Integer Programming optimization problem, while Chapter 5 introduces a constructive heuristic to efficiently allocate tenant applications in large-scale cloud platforms. The evaluation of the proposed strategy appears in Chapter 6 and, finally, Chapter 7 closes the thesis with final remarks and perspectives for future work.

2 LITERATURE REVIEW

Datacenters (DCs) are facilities consisting of tens to hundreds of thousands of servers (physical machines) and tens of hundreds of network devices (e.g., switches, routers and cables), power distribution systems and cooling systems. They can be divided in two classes: production and cloud datacenters. Production datacenters often have 100∼10k servers, present low rate of tenant arrival and departure and run data-analytics jobs from multiple groups/services. Cloud datacenters, in turn, usually have 50k∼300k servers, present high rate of tenant arrival and departure (churn), run both user-facing applications and inward computation and require elasticity (demands are more variable) (BENSON; AKELLA; MALTZ, 2010; BALLANI et al., 2011b; ABTS; FELDERMAN, 2012b).

Both classes of datacenters provide computational and network resources for tenants to run applications. Given the large scale of datacenters, the network is shared by a myriad of applications. However, the lack of control over how network resources are shared results in network performance variability and introduces vulnerabilities. This chapter is organized as follows. First, Section 2.1 presents an overview of datacenter networks. Then, Section 2.2 shows recent approaches and how they attempt to mitigate network resource sharing issues.

2.1 Datacenter Networks

A datacenter network (DCN) is the communication infrastructure used in datacenters and is described by the network topology, routing/switching equipment and the protocols (e.g., Ethernet, IP and TCP) (BARI et al., 2012). In this section, we show an overview of datacenter networks. First, Section 2.1.1 shows how DCNs are structured (i.e., the most common topology and some other topologies that have been recently proposed). Then, we discuss about network performance variability (Section 2.1.2) and network security (Section 2.1.3).

2.1.1 Topology

In this section, we present an overview of datacenter topologies. The network topology describes precisely how switches and hosts are interconnected. This is commonly represented as a graph in which vertices represent switches and hosts, and links are the edges that connect them.

Figure 2.1 shows a canonical three-tiered tree-like physical datacenter topology, which is implemented in today's datacenters. The datacenter topology three tiers are: *i*) the access (edge) layer, which consists of the Top-of-Rack (ToR) switches that connect the servers mounted on every rack; *ii*) the aggregation (distribution) layer, consisting of devices that interconnect the ToR switches in the access layer; and *iii*) the core layer, which consists of devices in the network core that interconnect the devices in the aggregation layer. Furthermore, every ToR switch may be connected to multiple aggregation switches for redundancy and every aggregation switch is connected to multiple core switches. Typically, a three-tiered network is implemented in datacenters with more than 8,000 servers (AL-FARES; LOUKISSAS; VAHDAT, 2008). In smaller datacenters, the core and aggregation layers are collapsed into one tier, resulting in a two-tiered datacenter topology (flat layer 2 topology) (BENSON; AKELLA; MALTZ, 2010).

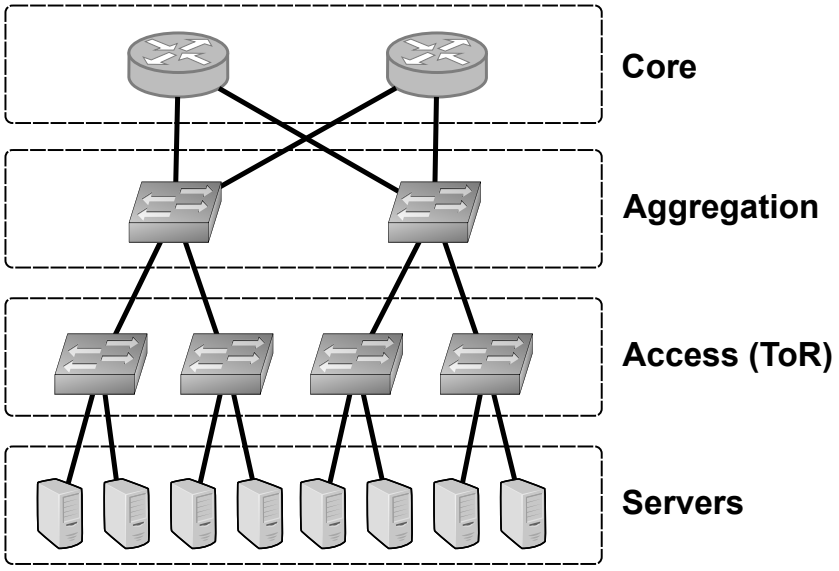


Figure 2.1: A canonical three-tiered tree-like datacenter network topology.

This multitiered topology has a significant amount of oversubscription, where servers attached to ToR switches have significantly more (possibly an order of magnitude more) provisioned bandwidth between one another than they do with hosts in other racks (ABTS; FELDERMAN, 2012b, 2012a). This is necessary in order to reduce network cost and improve resource utilization, which are key properties to help providers achieve economies of scale.

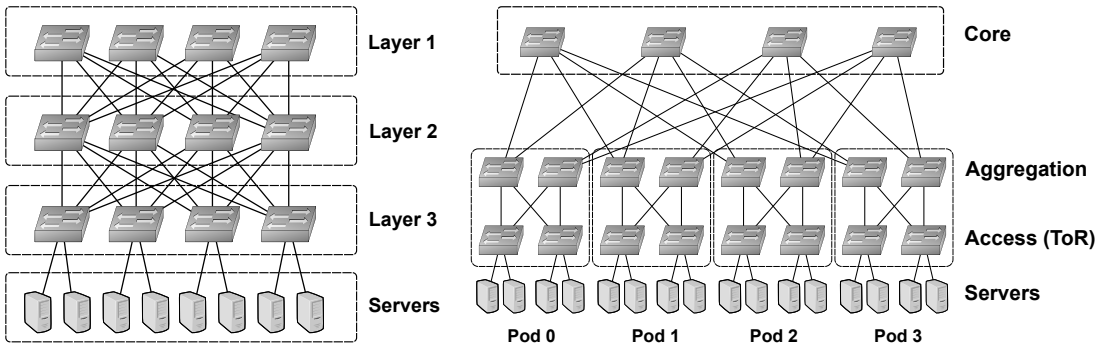
This topology, however, presents one major drawback: limited bisection bandwidth[1]. This limitation results in overloaded network core links (e.g., Microsoft researchers found links as much as 1:240 oversubscribed in their datacenters (GREENBERG et al., 2009)) and constrained server utilization (CURTIS; KESHAV; LOPEZ-ORTIZ, 2010; CURTIS et al., 2012).

[1]The bisection is the segmentation of a network in two equally-sized partitions. The sum of link capacities between the two partitions is called the bandwidth of the bisection. The bisection bandwidth of the network is the minimum bandwidth along all possible bisections. Typically, it refers to the worst-case segmentation of the network in two equal parts (FARRINGTON; RUBOW; VAHDAT, 2009).

To improve bisection bandwidth, some proposals follow a Clos-based (switch-oriented) design. A Clos topology is built up from multiple layers of switches (DALLY; TOWLES, 2003). Each switch in a layer is connected to all switches in the next layer, which provides extensive path diversity. Two proposals follow a Clos design: VL2 (GREENBERG et al., 2009, 2011) and Fat-Tree (AL-FARES; LOUKISSAS; VAHDAT, 2008).

VL2 (GREENBERG et al., 2009, 2011) is a Clos-based topology that provides a larger number of paths between any two aggregation switches than the conventional topology. This means that, for N intermediate switches, the failure of one of them reduces the bisection bandwidth by only 1/N, which the authors call graceful degradation of bandwidth. An example of VL2 is shown is Figure 2.2(a).

Fat-Tree (AL-FARES; LOUKISSAS; VAHDAT, 2008), in turn, is a specific type of Clos topology (folded Clos topology). The topology, shown in Figure 2.2(b), is organized in a k-ary tree-like structure. There are k pods, each one of them has two layers (aggregation and access) of k/2 switches. Each k/2 switch from the access layer is connected to k/2 servers and the remaining ports are connected to k/2 aggregation switches. Each of the $(k/2)^2$ k-port core switches has one port connected to each of k pods. More specifically, the i-th port of any core switch is connected to pod i so that consecutive ports in the aggregation layer of each pod switch are connected to core switches on k/2 strides. In general, a fat-tree built with k-port switches supports $k^3/4$ hosts.



(a) VL2. (b) Fat-Tree.

Figure 2.2: Clos-based topologies.

Other proposals for datacenter network topologies exist in the literature, such as server-oriented topologies (CamCube (ABU-LIBDEH et al., 2010)) and hybrid switch/server topologies (BCube (GUO et al., 2009) and Dcell (GUO et al., 2008)). Nevertheless, they are not easily implemented by providers because they require significant changes in comparison with today's topologies; thus, they have high implementation costs.

2.1.2 Performance Variability

Unlike computational resources (CPU, memory and storage), control of network usage is hindered because it is a distributed resource (SHIEH et al., 2011). Several papers (GIURGIU, 2010; MANGOT, 2009; SCHAD; DITTRICH; QUIANÉ-RUIZ, 2010; WANG; NG, 2010) concluded that VM bandwidth inside the cloud platform

(i.e., the throughput achieved by communication among pairs of VMs in the same datacenter) can vary by a factor of five or more. In particular, measurements of cloud resources (GIURGIU, 2010; LI et al., 2010; MANGOT, 2009; SCHAD; DITTRICH; QUIANÉ-RUIZ, 2010; WANG; NG, 2010) demonstrate that the intra-cloud network performance variability can negatively affect application execution time, resulting in poor and unpredictable overall application performance (BALLANI et al., 2011b; IOSUP; YIGITBASI; EPEMA, 2011; KOSSMANN; KRASKA; LOESING, 2010; ZAHARIA et al., 2008).

There are four factors that cause performance variability: type of traffic, heterogeneity of flows, network oversubscription and VM location (BALLANI et al., 2011b; SHIEH et al., 2011; ABTS; FELDERMAN, 2012b). These will be discussed next.

The type of traffic[2] in datacenters is one of the main causes for network performance variability. Despite the available mechanisms for TCP and its variants, such as DCCP and TFRC, to achieve scalability and high resource utilization, they do not provide robust performance isolation among flows from different applications (i.e., they allow flows to interfere with one another) (SHIEH et al., 2011). This scenario is further exacerbated by UDP, which does not have any mechanism to control how distinct flows share the network.

Second, the heterogeneity of flows (volume of data in each flow) inside a datacenter facilitates the occurrence of interference among flows, which is usually called *performance interference* (SHIEH et al., 2011; BARI et al., 2012). Flows are commonly classified as bimodal in the cloud: elephant or mice. Elephant flows have a large number of packets injected in the network over a short amount of time, exhibit bursty behavior and are long-lived. Mice flows, on their turn, have a small number of packets and are usually short-lived. The transient load imbalance created by elephant flows can adversely impact on any mice flow that is using the same heavily utilized link. For example, consider an elephant flow from A to B sharing a congested link with a mice flow from C to D. Any long-lived contention in the link increases the likelihood of discarding a packet from the C-to-D flow. Any packet discards will result in an unacknowledged packet at the sender and be retransmitted when the timeout period expires. Since the timeout period is generally one or two orders of magnitude more than the round-trip time (RTT), the latency may be a significant source of performance variability (ABTS; FELDERMAN, 2012b, 2012a).

The third factor, network oversubscription, exacerbates this scenario. The provider uses this factor to reduce operational cost, increase resource utilization and, thus, achieve economies of scale (MUDIGONDA et al., 2011). However, network traffic intensity varies over time and the momentary load imbalance resulting from this variation, which, at peak times, can be several times greater than the network capacity, creates contention, causing packets to be discarded (ABTS; FELDERMAN, 2012b).

Lastly, VM location inside the cloud infrastructure may be another source of performance variability. For instance, communication between a pair of VMs on the

[2]Traffic is tipically identified by flows in datacenters. Flows are characterized by sequences of packets from a source to a destination host.

same rack has higher performance predictability than communication between pairs of VMs located on opposite ends of the network topology. This happens because the first one is subject to interference originated from a smaller quantity of flows in the network than the second one; in the second, the flow needs to traverse links closer to the network core and, therefore, shared by a larger number of applications (BALLANI et al., 2011b; SCHAD; DITTRICH; QUIANÉ-RUIZ, 2010).

2.1.3 Security

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 (SHIEH et al., 2011, 2010; RISTENPART et al., 2009).

In particular, Liu (2010), Shieh et al. (2011) and Ristenpart et al. (2009) show intra-cloud network vulnerabilities. Liu (2010) shows how an adversary can obtain information about the cloud infrastructure in order to launch DoS attacks, while Shieh et al. (2011) show how to launch performance interference attacks. These attacks are made possible because of four main reasons. First, as previously discussed, TCP cannot provide robust isolation among flows. Second, DCNs are usually under-provisioned (i.e., oversubscribed), since it is expensive to build a network with full bisection bandwidth (CURTIS; KESHAV; LOPEZ-ORTIZ, 2010; CURTIS et al., 2012). Third, a cloud datacenter is used by many clients (organizations and common users), which opens up the possibility for adversaries to attack other applications hosted in the same datacenter. Fourth, a tenant usually has little or no control over the network.

This way, selfish or malicious parties can simply saturate the limited network bandwidth from a bottleneck link to perform performance interference or DoS attacks against other applications in the DCN. For instance, an adversary can degrade network performance of targeted victims (performance interference) or, by using a variable number of flows or higher rate UDP flows, can utilize all of the available network bandwidth (DoS) (SHIEH et al., 2011). Liu (2010) describes two types of DoS: untargeted and targeted attacks. In an untargeted attack, an adversary wants to stop network communication for some VMs in any portion of the DCN. Targeted ones, on the other hand, are directed for specific VMs (or applications) in the cloud.

Ristenpart et al. (2009), on their turn, conducted a case study in Amazon Elastic Compute Cloud (EC2). The authors report they have no privileged information, that is, they know only the information publicly disclosed by Amazon. They assume that attackers are common cloud customers (i.e., they can perform the same actions of other users). First, the internal cloud infrastructure is mapped in accordance with its availability zones in each region. With this mapping, they concluded that IP addresses are statically partitioned between regions, which facilitates management of routing, but helps a malicious user.

After successfully mapping EC2, the next step of an attack is to determine co-residency between VMs. They identify several network-based potential methods for checking if two VM instances are co-resident. Instances are likely co-resident if one or more of the following tests is true:

- *Matching Dom0 IP address.* Since Amazon uses Xen (BARHAM et al., 2003) virtualization technology, an instance network traffic first hop is the Dom0 privileged VM. Tenants can determine their Dom0 IP from the first hop on any route out from the instance. Similarly, a user determines an uncontrolled instance Dom0 IP address by verifying the last hop in a TCP SYN traceroute.

- *Small packet round-trip times.* They verify that round-trip times for traffic between co-resident VMs are smaller than RTTs between VMs hosted by distinct physical servers.

- *Numerically close internal IP addresses (e.g., within 7).* The authors state that, according to their experiments, the same Dom0 IP is shared by instances with a contiguous sequence of internal IP addresses.

Based on the information acquired, Ristenpart et al. (2009) show that malicious users can exploit VM placement in EC2 to colocate their VMs in the same server as the target victims. Thus, they can take advantage of the shared network to carry out several kinds of attacks, such as man-in-the-middle, extraction of confidential information, performance interference and DoS.

To sum up, attacks are made easier because providers lack mechanisms to perform robust isolation among tenants and to manage bandwidth used by VMs for intra-cloud communication (POPA et al., 2012; XIE et al., 2012; GUO et al., 2010a).

2.2 Cloud Network Resource Sharing

Providers use VLANs (IEEE COMPUTER SOCIETY, 2005) 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 (GREENBERG et al., 2009) and Fat-Tree (AL-FARES; LOUKISSAS; VAHDAT, 2008)) (GUO et al., 2010b). Second, VLANs do not provide bandwidth guarantees.

Current resource allocation algorithms employed by public cloud providers do not handle network performance and security (KITSOS et al., 2012). Such algorithms use round-robin across servers or across racks, taking into account only computational resources (processing power, memory and storage). Nevertheless, allocation algorithms must also be aware of network resources, in order to protect tenant data in the network.

Some recent work seek to provide efficient means of controlling network resource sharing, in order to improve network predictability and security for tenants. We divide these proposals in three classes: virtual machine consolidation (Section 2.2.1), proportional sharing (Section 2.2.2) and network virtualization (Section 2.2.3). Finally, we discuss the presented approaches in Section 2.3.

2.2.1 Virtual Machine Consolidation

Some recent work handles the problem of network-aware resource allocation as VM consolidation (i.e., VM placement). Meng, Pappas & Zhang (2010) propose a traffic-aware VM allocation process by adapting the Stochastic Bin Packing optimization problem (KLEINBERG; RABANI; TARDOS, 1997; GOEL; INDYK, 1999). Wang, Meng & Zhang (2011) capture VM bandwidth demand with random variables that follow probabilistic distributions and propose an approximation algorithm to efficiently solve the problem. Breitgand & Epstein (2012), in turn, present an approximation algorithm closer to the optimum solution than Wang, Meng & Zhang (2011).

HomeAlone (ZHANG et al., 2011), from another perspective, explores vulnerabilities of physical co-residence of VMs to increase the security of computational resources. That is, the key idea is to invert the usual application of side channels. Rather than exploiting a side channel as a vector of attack, HomeAlone uses a side-channel as a defensive detection tool. By analyzing cache usage during periods in which well-behaved VMs coordinate to avoid portions of the cache, a tenant can detect the activity of a co-resident malicious VM.

2.2.2 Proportional Sharing

Lam et al. (2012) and Shieh et al. (2010, 2011) focus on providing fair network sharing in accordance with weights assigned to VMs or tenants. NetShare (LAM et al., 2012) proposes a statistical multiplexing mechanism to allocate bandwidth for tenants in a proportional way, to achieve high link utilization and to provide weighted hierarchical max-min fair sharing (bandwidth unused by an application is shared proportionally by other applications). The weights are either specified by a manager or automatically assigned at each switch port based on a virtual machine heuristic.

Seawall (SHIEH et al., 2010, 2011), in turn, is a bandwidth allocation scheme that divides network capacity based on an administrator-specified policy. The key ideia is to assign weights to any entity that generates traffic (such as a VM or a process) and to allocate bandwidth according to these weights in a proportional way. It uses congestion control and point to multipoint tunnels to enforce bandwidth sharing policies.

2.2.3 Network Virtualization

Another class of approaches proposes to allocate each application (or tenant) in a distinct virtual network. The most representative papers of this class are: SecondNet (GUO et al., 2010b, 2010a), CloudNaaS (BENSON et al., 2011), Gatekeeper (RODRIGUES et al., 2011), Oktopus (BALLANI et al., 2011b) and Proteus (XIE et al., 2012).

SecondNet proposes the allocation of Virtual Datacenters (VDCs) as a means to provide bandwidth guarantees for pairs of VMs in a multi-tenant datacenter. It defines three service types: (a) guaranteed bandwidth between pairs of VMs (type

0); (*b*) bandwidth guarantees for the first and/or last hops of a path (type 1); and (*c*) a best-effort service (type 2).

Gatekeeper focuses on providing guaranteed bandwidth among VMs in a datacenter and achieving a high network utilization. This is done by defining both the minimum and maximum allowed bandwidth for each VM. CloudNaaS, in turn, is a virtual network architecture to deploy applications in cloud platforms with a set of network functions (e.g., custom addressing and service differentiation).

Oktopus introduces two virtual network abstractions, namely virtual cluster (VC) and virtual oversubscribed cluster (VOC), in order to explore the trade-off among performance guarantees offered to tenants, their costs, and the provider revenue. Proteus, on its turn, attempts to address the low resource utilization caused by Oktopus by proposing the Time-Interleaved Virtual Cluster (TIVC) abstraction, which allows specifying the time-varying networking requirements of cloud applications.

2.3 Discussion

In this section, we provide a detailed discussion about the previously presented projects. Each class (VM consolidation, proportional sharing and network virtualization) introduces some advantages and disadvantages. In particular, some drawbacks, such as resource management overhead and underutilization of resources, may hinder the implementation of the approach in cloud platforms because they hurt fundamental properties of the cloud computing paradigm (e.g., scalability).

VM consolidation approaches present strategies and mathematical models directed to computational resources (servers and virtual machines) over network resources, aiming at reducing the number of servers used during the process of resource allocation. Nevertheless, the network must be taken into account by the allocation process when providing security and network-performance predictability for tenants.

Proportional sharing provides fair network sharing among entities. However, it adds substantial management overhead to control each VM network share (especially in large-scale shared infrastructures), wasting network resources. In particular, NetShare may not be well-suited for cloud datacenters for two reasons. First, scalability is compromised because queues have to be configured at each switch port for each application. Second, it relies on specific features of Fulcrum switches to implement its mechanisms, which reduces its deployability (BARI et al., 2012).

Network virtualization proposals isolate one tenant (or application) per virtual network. Nonetheless, they present three main 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 (ABTS; FELDERMAN, 2012b); *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) (SHIEH et al., 2011). These drawbacks hurt provider revenue and, ultimately, translate to higher tenant costs.

More specifically, SecondNet and Proteus present the following drawbacks. SecondNet results in a clique virtual topology (with dense connectivity), which makes it difficult for the provider to multiplex multiple tenants on the underlying network infrastructure (BALLANI et al., 2011b). Proteus, in turn, results in high allocation complexity and high implementation costs.

To sum up, cloud resource sharing presents two shortcomings: *i*) network resources are scarce and often represent the bottleneck when compared to computational resources in datacenters (XIE et al., 2012; GREENBERG et al., 2009); and *ii*) the lack of isolation provide means for malicious parties to launch attacks against well-behaved tenants. These attacks will be further discussed in Chapter 3.

3 FOUNDATIONS

In this chapter, we define the basis of our approach. More specifically, we present the threat model considered (Section 3.1) and the system model (Section 3.2). These fundamental definitions are used throughout the rest of this thesis.

3.1 Threat Model

We consider an IaaS tenant that operates one or more applications[1]. Each tenant has the same privileges as the others, similarly to Ristenpart et al. (2009). 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 (SHIEH et al., 2011). 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 (RISTENPART et al., 2009) to collocate their VMs near the target.

Attacks against the availability of network resources are performed in two ways: *i*) by increasing the volume of data in each flow and the number of flows in the network, exploiting the lack of traffic isolation among applications (XIE et al., 2012); and *ii*) by sending large UDP flows. Since all tenants share the same network (they compete for bandwidth in the intra-cloud network), such attacks are not limited to their targets, but rather can affect many applications. Therefore, a tenant with many VMs can cumulatively send enough traffic to overflow the receiver, some network link en route to that host or other network bottleneck. Furthermore, TCP only achieves long-term throughput fairness, which may lead to low-RTT flows getting higher shares of the bandwidth than high-RTT flows (PRAKASH et al., 2012). Figure 3.1 shows an example of performance interference attack (that may even lead to DoS), in which an attack can take place by compromising a link between two virtual switches.

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, ac-

[1]Basically, each application consists of a collection of VMs. We will define an application in Section 3.2.1.

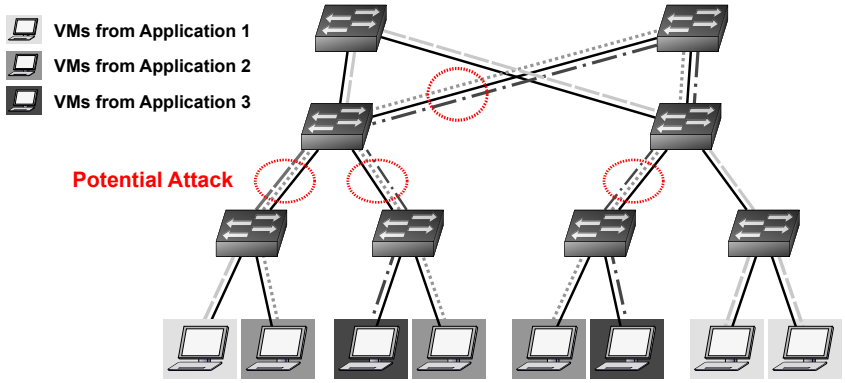


Figure 3.1: Potential attack against network availability due to the lack of mechanisms to control network resource sharing.

counts can be easily obtained (no strict requirements for accounts), or compromised (user behavior) (GREENBERG et al., 2008). Furthermore, the detection of such attacks is not simple because of two reasons. First, an attacker can conceal malicious traffic as well-behaved (SHIEH et al., 2011). Second, a persistent attacker is not easily deterred by obfuscation schemes (SHIEH et al., 2011), i.e., techniques used by providers to hide resource location, for instance against network-based VM co-residence checks and internal cloud infrastructure mapping (RISTENPART et al., 2009).

3.2 System Model

In this section, we define fundamental formulations used to model our strategy. The notation is 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. Table 3.1 shows an overview of the symbols defined in this section. The notation is similar to that employed in Chowdhury, Rahman & Boutaba (2009).

3.2.1 Application Requests

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

A request also specifies network requirements. Tenants are provided 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 3.2. This abstraction is motivated by the observation that, in private environments, tenants typically run their applications on dedicated clusters, with computational nodes connected

Table 3.1: Notation of cloud resource allocation problem.

Symbol	Description
A^r	Set of applications.
M_a^r	Number of requested VMs for application $a \in A^r$.
$Band_a^r$	Bandwidth requested for each VM of application $a \in A^r$.
T_{a_i,a_j}^r	Trust relationship between applications a_i and a_j.
I^v	Set of virtual infrastructures.
S_i^v	Set of virtual switches in $i \in I^v$.
R_i^v	Set of virtual Top-of-Rack switches in $i \in I^v$.
M_i^v	Set of VMs in $i \in I^v$.
E_i^v	Set of virtual links in $i \in I^v$.
$Band^v(e^v)$	Bandwidth of virtual link $e^v \in E_i^v$.
$Oversub^v(e^v)$	Oversubscription factor of virtual link $e^v \in E_i^v$.
S^s	Set of physical switches.
R^s	Set of physical Top-of-Rack switches.
M^s	Set of servers.
E^s	Set of physical links.
$Band^s(e^s)$	Bandwidth of physical link $e^s \in E^s$.
$Slots^s(m^s)$	Number of slots for VMs of server $m^s \in M^s$.
$Cost^s(s^s)$	Cost of $s^s \in S^s$ to host a virtual switch.
$Cap^s(s^s)$	Maximum number of virtual switches that $s^s \in S^s$ can host.

through Ethernet switches (BALLANI et al., 2011b).

Trust relationships between applications are represented by T_{a_i,a_j}^r, which denotes whether application a_i from a 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 (ZIMMERMANN, 1995). 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.

3.2.2 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 $G_i^v = < S_i^v, R_i^v, M_i^v, E_i^v, Band^v, Oversub^v >$. The set of network

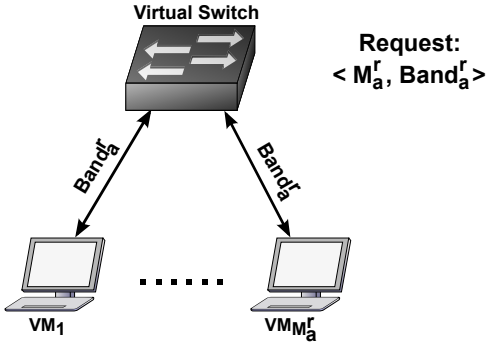


Figure 3.2: Tenant's view of an application's network topology.

devices in i is denoted by S_i^v and is composed by switches only, similarly to Ballani et al. (2011b) and Xie et al. (2012). 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 $e^v = (u, w) \in E_i^v \mid u \neq w$ 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 bandwidth $Band^v(e^v) \in \mathbb{R}^+$ and an oversubscription factor $Oversub^v(e^v) \in \mathbb{R}$ employed by the provider to increase network resource utilization.

3.2.3 Physical Infrastructure

The physical substrate is composed of servers, network devices and links, similarly to Guo et al. (2010a). This infrastructure is represented as a weighted bidirectional graph $G^s =< S^s, R^s, M^s, E^s, Band^s, Slots^s, Cost^s, Cap^s >$, 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), because the closer the switch is to the network core, the more oversubscribed it tends to be. The subset of ToR switches is represented by $R^s \subset S^s$. Each link, in turn, $e^s = (u, w) \in E^s \mid u, w \in S^s \cup M^s$ and $u \neq w$ between nodes u and w is associated with a 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.

4 RESOURCE ALLOCATION STRATEGY

This chapter presents, using the model defined in Chapter 3, a novel approach to allocate resources for incoming application requests at cloud platforms. Unlike previous approaches, the one proposed in this thesis takes both security and network performance into account. The 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 expected traffic matrix between VMs of the same application.

To provide security- and network-performance-aware resource allocation, there is a fundamental challenge to be addressed: the mapping of resources considering bandwidth-constrained network links is an NP-Hard problem (GUO et al., 2010b).

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 4.1, which contains three functions. 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 we expect $\mathcal{G}$ to be used first to allocate the VIs on the cloud substrate whereas $\mathcal{F}$ can be used later to allocate every incoming application request.

Table 4.1: Defined functions for cloud resource allocation.

Notation	**Description**
Function $\mathcal{H}$	Maps applications directly to the cloud physical substrate.
Function $\mathcal{G}$	Allocates the set of virtual infrastructures onto the physical substrate of the cloud platform.
Function $\mathcal{F}$	Maps each application request into a virtual infrastructure according to security and performance criteria.

The proposed approach takes as input a set of virtual infrastructures. The proper

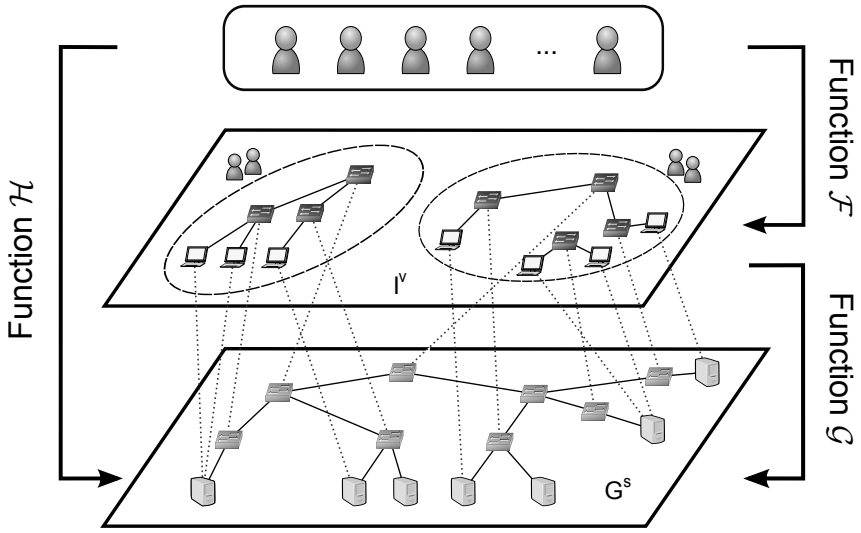


Figure 4.1: Cloud resource allocation overview.

definition about the number of VIs and their sizes affects the efficiency of functions $\mathcal{F}$ and $\mathcal{G}$. This could be determined arbitrarily by the provider, or for example based on the resource utilization history (MENG; PAPPAS; ZHANG, 2010), with information from already allocated applications collected by tools such as Amazon CloudWatch[1]. Furthermore, the network topology of each VI is not restricted by the physical infrastructure topology (CHOWDHURY; RAHMAN; BOUTABA, 2009), since the heterogeneity of topologies is addressed by function $\mathcal{G}$. Further investigation on mechanisms for choosing VIs is out of scope and left for future work.

The next sections describe functions $\mathcal{G}$ and $\mathcal{F}$, respectively. Lastly, Section 4.3 shows the optimal resource allocation evaluation.

4.1 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) (CHOWDHURY; RAHMAN; BOUTABA, 2009), 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 3.2.2) and the physical substrate (Section 3.2.3). Furthermore, three parameters are used: $\alpha_{(w_1,w_2)}$, γ and δ, defined as follows. Factor $\alpha_{(w_1,w_2)}$ quantifies the importance of link (w_1, w_2) within the $(0, 1]$ range. Parameters γ and δ balance both components of the objective function [Equation (4.1)].

Variables. The variables related to this function are:

- $x_{i,s^v,s^s} \in \{0, 1\}$: indicates if virtual switch $s^v \in S_i^v$ from virtual infrastructure

[1] `http://aws.amazon.com/cloudwatch/`

$i \in I^v$ is allocated on physical switch $s^s \in S^s$;

- $y_{i,e^v,(w_1,w_2)} \in \{0,1\}$: indicates if virtual link $e^v \in E_i^v$, which belongs to virtual infrastructure $i \in I^v$, uses physical link $(w_1, w_2) \in E^s$;
- $z_{i,m^v,m^s} \in \{0,1\}$: indicates if VM $m^v \in M_i^v$ from virtual infrastructure $i \in I^v$ is allocated at server $m^s \in M^s$.

Objective. Equation (4.1) minimizes the amount of network resources used to allocate the virtual infrastructures. 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. The second component quantifies the cost of allocating virtual switches on physical switches.

$$Z = \text{Min } \gamma * \sum_{(w_1,w_2) \in E^s} \frac{\alpha_{(w_1,w_2)}}{Band^s(w_1,w_2)} * \left(\sum_{i \in I^v} \sum_{e^v \in E_i^v} y_{i,e^v,(w_1,w_2)} * Band^v(e^v) \right) + \delta * \sum_{s^s \in S^s} \sum_{i \in I^v} \sum_{s^v \in S_i^v} x_{i,s^v,s^s} * Cost^s(s^s) \quad (4.1)$$

The set of constraints guides 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 $\mathcal{M}_n : (M_i^v \cup S_i^v) \rightarrow (M^s \cup S^s)$[2], $\forall i \in I^v$ 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 both 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.

[2]The set of switches (S_i^v) is composed by all network devices, including ToR switches (R_i^v), as we formally defined in Chapter 3.

Constraints. We present the model constraints in the following manner. Constraint sets (4.2) and (4.3) ensure that one VM and one virtual switch are mapped to one physical machine and one physical switch, respectively. Additionally, Equation (4.4) assures that virtual switches from the same VI are allocated in distinct physical switches.

$$\sum_{m^s \in M^s} z_{i,m^v,m^s} = 1 \qquad \forall i \in I^v \quad \forall m^v \in M_i^v \quad (4.2)$$

$$\sum_{s^s \in S^s} x_{i,s^v,s^s} = 1 \qquad \forall i \in I^v, \quad \forall s^v \in S_i^v \quad (4.3)$$

$$\sum_{i \in I^v} \sum_{s^v \in S_i^v} x_{i,s^v,s^s} \leq 1 \qquad \forall i \in I^v, \quad \forall s^s \in S^s \quad (4.4)$$

Constraint sets (4.5), (4.6) and (4.7) guarantee that physical resource capacity (from servers, switches and links, respectively) is not exceeded. This is achieved by verifying if the sum of the allocated capacity for virtual resources is not superior to the capacity of the physical resource in which they are mapped.

$$\sum_{i \in I^v} \sum_{m^v \in M_i^v} z_{i,m^v,m^s} \leq Slots^s(m^s) \qquad \forall m^s \in M^s \quad (4.5)$$

$$\sum_{i \in I^v} \sum_{s^v \in S_i^v} x_{i,s^v,s^s} \leq Cap^s(s^s) \qquad \forall s^s \in S^s \quad (4.6)$$

$$\sum_{i \in I^v} \sum_{e^v \in E_i^v} (y_{i,e^v,(w_1,w_2)} * Band^v(e^v)) \leq Band^s(e^s) \qquad \forall e^s = (w_1, w_2) \in E^s \quad (4.7)$$

The last constraint sets are related to virtual link embedding. Equation (4.8) maps virtual links to valid substrate paths (unsplittable flows) between the corresponding substrate switches that host the end virtual switches of that virtual link. It compares the in-degree and the out-degree of each physical switch w_3.

$$\sum_{\substack{w_4 \in S^s \setminus \{w_3\} | \\ (w_3,w_4) \in E^s}} y_{i,e^v,(w_3,w_4)} - \sum_{\substack{w_4 \in S^s \setminus \{w_3\} | \\ (w_4,w_3) \in E^s}} y_{i,e^v,(w_4,w_3)} = x_{i,w_1,w_3} - x_{i,w_2,w_3}$$
$$\forall i \in I^v, \quad \forall w_3 \in S^s, \quad \forall w_1, w_2 \in S_i^v \mid e^v = (w_1, w_2) \in E_i^v \quad (4.8)$$

Equations (4.9) and (4.10), in turn, map virtual links between VMs and ToR switches. More specifically, Equation (4.9) guarantees that there is an outgoing link to connect each virtual switch to each one of its VMs, while Equation (4.10) ensures that there is an incoming link to each VM from its ToR switch.

$$x_{i,s^v,s^s} = \sum_{m^s \in M^s | (m^s,s^s) \in E^s} y_{i,e^v,(m^s,s^s)}$$
$$\forall i \in I^v, \quad \forall s^s \in S^s, \quad \forall m^v \in M_i^v, \quad \forall s^v \in R_i^v \mid e^v = (s^v, m^v) \in E_i^v \quad (4.9)$$

$$z_{i,m^v,m^s} = y_{i,e^v,(m^s,s^s)} \quad \forall i \in I^v, \quad \forall m^s \in M^s,$$
$$\forall m^v \in M_i^v, \forall s^s \in S^s \mid (s^s, m^s) \in E^s, \forall s^v \in R_i^v \mid e^v = (s^v, m^v) \in E_i^v \quad (4.10)$$

Finally, Equation (4.11) ensures that links are symmetric. That is, if there is a link from one virtual node $n_1 \in \{M_i^v \cup S_i^v\}$ to another node $n_2 \in \{M_i^v \cup S_i^v\}$, then there is a link from n_2 to n_1.

$$y_{i,e_1^v,(w_1,w_2)} - y_{i,e_2^v,(w_2,w_1)} = 0$$
$$\forall i \in I^v, \forall (w_1, w_2) \in E^s, \quad \forall e_1^v = (w_3, w_4) \in E_i^v \mid e_2^v = (w_4, w_3) \in E_i^v \quad (4.11)$$

4.2 Mapping Applications into Virtual Infrastructures

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

Input. This function receives as input an incoming application request (Section 3.2.1), the set of virtual infrastructures (Section 3.2.2), 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 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 variables related to this function are:

- $g_{a,i,r,m} \in \{0, 1\}$: indicates if VM $m \in M_i^v$ at rack $r \in R_i^v$ from VI $i \in I^v$ was allocated for application $a \in A^r$;
- $G_{a,i} \in \{0, 1\}$: indicates if application $a \in A^r$ is located at VI $i \in I^v$;
- $h_{a,i,(w_1,w_2),p} \in \{0, 1\}$: indicates if application $a \in A^r$ uses link $(w_1, w_2) \in E_i^v \mid w_1, w_2 \in S_i^v$ for communication between the pair of racks $p = (r_1, r_2) \in P_i^v$ from VI $i \in I^v$;

- $H_{i,a_1,a_2} \in \{0,1\}$: indicates whether applications a_1 and a_2 are allocated at VI $i \in I^v$;

- $B_{a,i,p} \in \{0,1\}$: indicates if application a needs bandwidth for communication between the pair of racks $p \in \mathcal{P}_i^v$ 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 (4.13)].

Objective. Equation (4.12) addresses two key 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 in two ways. 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.

$$Z = \text{Min } \gamma * \left(\sum_{i \in I^v} \sum_{a_1 \in A^r} \sum_{a_2 \in A^r} (1 - T^r_{a_1,a_2}) * H_{i,a_1,a_2} \right) + \delta \left(\sum_{a \in A^r} \sum_{i \in I^v} \sum_{(w_1,w_2) \in E_i^v} \sum_{p \in \mathcal{P}_i^v} F_{a,i,w_1,w_2,p} \right) \quad (4.12)$$

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 bandwidth consumption by employing the concept of VM clusters[3]. 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. Using less resources on average when accommodating requests allows more requests to be accepted.

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

[3]This concept is similar to that of VM grouping, used in Ballani et al. (2011b). We prefer the term *VM cluster* so to avoid confusion with application grouping.

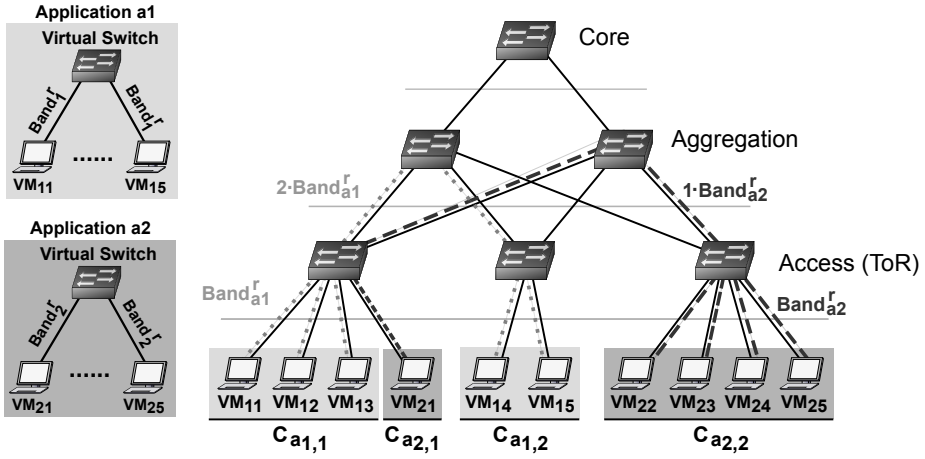


Figure 4.2: Communication between VM clusters.

available bandwidth for communication between these clusters. For instance, consider the scenario presented in Figure 4.2, where two applications have two VM clusters each: 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 $Band_{a_1}^r$, traffic between the pair of clusters $C_{a_1,1}$ and $C_{a_1,2}$ is limited by the cluster with the lowest rate: $\min(|C_{a_1,1}|, |C_{a_1,2}|) * Band_{a_1}^r$. Thus, the bandwidth required by one VM cluster to communicate with all other clusters of the same application is given by the following expression:

$$B_{a_x,c_i} = \min\left(|c_i| * Band_{a_x}^r, \sum_{c \in C_{a_x}^r, c \neq c_i} |c| * Band_{a_x}^r\right) \qquad \forall c_i \in C_{a_x}^r \quad (4.13)$$

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

Path selection. The model presented in this thesis considers the use of a single path for communication between pairs of VM clusters from the same application. The use of a single path does not limit the proposed approach for the following reason. Datacenters typically have networks with rich connectivity, such as multi-rooted trees (MUDIGONDA et al., 2010) and Fat-Tree (AL-FARES; LOUKISSAS; VAHDAT, 2008). Cloud network devices are usually connected with several links that are balanced by multipathing techniques, such as Equal-Cost Multi-Path (ECMP) (ABTS; FELDERMAN, 2012b) and Valiant Load Balancing (VLB) (XIE et al., 2012). 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 (BALLANI et al., 2011b).

Constraints. We present the model constraints in the following manner. First, the model quantifies the number of mutually untrusted relationships between tenant applications allocated in the same VI by means of the value assigned to $H_{a_1,a_2,i}$ $\forall a_1,a_2 \in A^r$, $\forall i \in I^v$. The value of H would be defined in the following constraint:

$$G_{a_1,i} * G_{a_2,i} = H_{a_1,a_2,i} \qquad \forall a_1, a_2 \in A^r \mid a_1 \neq a_2, \quad \forall i \in I^v \quad (4.14)$$

However, this equation represents a quadratic constraint, which cannot be employed in a linear optimization model. Therefore, we linearize this constraint by creating three new constraint sets [Equations (4.15), (4.16) and (4.17)] to indicate if a pair of applications is allocated in the same VI.

$$H_{a_1,a_2,i} \leq G_{a_1,i} \qquad \forall a_1, a_2 \in A^r \mid a_1 \neq a_2, \quad \forall i \in I^v \quad (4.15)$$

$$H_{a_1,a_2,i} \leq G_{a_2,i} \qquad \forall a_1, a_2 \in A^r \mid a_1 \neq a_2, \quad \forall i \in I^v \quad (4.16)$$

$$H_{a_1,a_2,i} \geq G_{a_1,i} + G_{a_2,i} - 1 \qquad \forall a_1, a_2 \in A^r \mid a_1 \neq a_2, \quad \forall i \in I^v \quad (4.17)$$

Constraint set (4.18) defines the value of the variable indicating in which VI the application is allocated. The value of $G_{a,i}$ is set to 1 if and only if all VMs from application a are allocated at VI $i \in I^v$. Otherwise, it is set to 0.

$$G_{a,i} = \frac{\sum_{r \in R_i^v} \sum_{m \in M_i^v \mid (r,m) \in E_i^v} g_{a,i,r,m}}{M_a^r} \qquad \forall a \in A^r, \quad \forall i \in I^v \quad (4.18)$$

The upcoming constraint sets specify how applications are mapped in VIs according to the available resources. Constraint sets (4.19) and (4.20) ensure that all VMs requested by each application are allocated and that all VMs from the same application are allocated in the same VI, respectively.

$$\sum_{i \in I^v} \sum_{r \in R_i^v} \sum_{\substack{m \in M_i^v \mid \\ (r,m) \in E_i^v}} g_{a,i,r,m} = M_a^r \qquad \forall a \in A^r \quad (4.19)$$

$$g_{a,i,r,m} \leq G_{a,i} \qquad \forall a \in A^r, \quad \forall i \in I^v, \quad \forall r \in R_i^v, \quad \forall m \in M_i^v \mid (r,m) \in E_i^v \quad (4.20)$$

Constraint set (4.21) guarantees that one VM is only allocated to one application, while constraint set (4.22) ensures that link capacity is not exceeded (i.e., the sum of all bandwidth allocated at link e^v cannot be superior to its capacity).

$$\sum_{a \in A^r} g_{a,i,r,m} \leq 1 \qquad \forall i \in I^v, \quad \forall r \in R_i^v, \quad \forall m \in M_i^v \mid (r,m) \in E_i^v \quad (4.21)$$

$$\sum_{a \in A^r} \sum_{p \in P_i^v} F_{a,i,(w_1,w_2),p} \leq Band^v(e^v) * Oversub^v(e^v)$$
$$\forall i \in I^v, \quad \forall e^v = (w_1, w_2) \in E_i^v \quad (4.22)$$

Finally, the remaining constraints are related to flow conservation. Equations (4.23) and (4.24) calculate the outgoing and the incoming traffic from the source and destination racks, respectively. The value is calculated according to what was previously explained in Equation (4.13).

$$\sum_{w_2 \in S_i^v | (r_1,w_2) \in E_i^v} F_{a,i,(r_1,w_2),p} = \min \left(\sum_{m \in M_i^v} g_{a,i,r_1,m}, \sum_{m \in M_i^v} g_{a,i,r_2,m} \right) * Band_a^r$$
$$\forall i \in I^v, \quad \forall a \in A^r, \quad \forall p = (r_1, r_2) \in P_i^v \quad (4.23)$$

$$\sum_{w_1 \in S_i^v | (w_1,r_2) \in E_i^v} F_{a,i,(w_1,r_2),p} = \min \left(\sum_{m \in M_i^v} g_{a,i,r_1,m}, \sum_{m \in M_i^v} g_{a,i,r_2,m} \right) * Band_a^r$$
$$\forall i \in I^v, \quad \forall a \in A^r, \quad \forall p = (r_1, r_2) \in P_i^v \quad (4.24)$$

However, Equations (4.23) and (4.24) are not linear. Therefore, we linearize these constraint sets by creating seven new linear sets [Equations (4.25) to (4.31)]. First of all, we introduce an auxiliary variable $B_{a,i,p} \in \{0, 1\}$ (which was previously defined). Considering that $M_a^r * Band_a^r$ is an upper bound for $\sum_{m \in M_i^v | (r_1,m) \in E_i^v} g_{a,i,r_1,m} * Band_a^r$ and for $\sum_{m \in M_i^v | (r_2,m) \in E_i^v} g_{a,i,r_2,m} * Band_a^r$, we define constraints sets (4.25) and (4.26) to get whichever value produces the smaller value of $\sum_{w_2 \in S_i^v | (r_1,w_2) \in E_i^v} F_{a,i,(r_1,w_2),p}$.

$$\sum_{w_2 \in S_i^v | (r_1,w_2) \in E_i^v} F_{a,i,(r_1,w_2),p} \geq \sum_{m \in M_i^v | (r_1,m) \in E_i^v} g_{a,i,r_1,m} * Band_a^r - (M_a^r * Band_a^r * B_{a,i,p})$$
$$\forall i \in I^v, \quad \forall a \in A^r, \quad \forall p = (r_1, r_2) \in P_i^v \quad (4.25)$$

$$\sum_{w_2 \in S_i^v | (r_1,w_2) \in E_i^v} F_{a,i,(r_1,w_2),p} \geq \sum_{m \in M_i^v | (r_2,m) \in E_i^v} g_{a,i,r_2,m} * Band_a^r - (M_a^r * Band_a^r * (1 - B_{a,i,p}))$$
$$\forall i \in I^v, \quad \forall a \in A^r, \quad \forall p = (r_1, r_2) \in P_i^v \quad (4.26)$$

Since variable $F_{a,i,(r_1,w_2),p}$ is used in other constraint sets, its value is not guaranteed to be the minimum between both VM clusters. Thus, we need additional

formulations to ensure that this variable will have the correct value (i.e., the minimum bandwidth that satisfies the VM clusters at racks r_1 and $r_2 \in R_i^v \quad \forall i \in I^v$). Equations (4.27) and (4.28) compute which is the smaller VM cluster.

$$\sum_{m \in M_i^v | (r_1,m) \in E_i^v} g_{a,i,r_1,m} \leq \sum_{m \in M_i^v | (r_2,m) \in E_i^v} g_{a,i,r_2,m} * Band_a^r + (M_a^r * Band_a^r * B_{a,i,p}) \quad \forall i \in I^v, \quad \forall a \in A^r, \quad \forall p = (r_1, r_2) \in P_i^v \quad (4.27)$$

$$\sum_{m \in M_i^v | (r_2,m) \in E_i^v} g_{a,i,r_2,m} \leq \sum_{m \in M_i^v | (r_1,m) \in E_i^v} g_{a,i,r_1,m} * Band_a^r + (M_a^r * Band_a^r * (1 - B_{a,i,p})) \quad \forall i \in I^v, \quad \forall a \in A^r, \quad \forall p = (r_1, r_2) \in P_i^v \quad (4.28)$$

Constraint sets (4.29) and (4.30) ensure that $\sum_{w_2 \in S_i^v | (r_1,w_2) \in E_i^v} F_{a,i,r_1,w_2,p}$ and $\sum_{w_2 \in S_i^v | (r_1,w_2) \in E_i^v} F_{a,i,r_1,w_2,p}$ will receive the maximum allowed value (that is, the minimum bandwidth required for communication among VM clusters from application a allocated at racks r_1 and r_2).

$$\sum_{w_2 \in S_i^v | (r_1,w_2) \in E_i^v} F_{a,i,r_1,w_2,p} \leq \sum_{m \in M_i^v | (r_1,m) \in E_i^v} g_{a,i,r_1,m} * Band_a^r + (M_a^r * Band_a^r * B_{a,i,p}) \quad \forall i \in I^v, \quad \forall a \in A^r, \quad \forall p = (r_1, r_2) \in P_i^v \quad (4.29)$$

$$\sum_{w_2 \in S_i^v | (r_1,w_2) \in E_i^v} F_{a,i,r_1,w_2,p} \leq \sum_{m \in M_i^v | (r_2,m) \in E_i^v} g_{a,i,r_2,m} * Band_a^r + (M_a^r * Band_a^r * (1 - B_{a,i,p})) \quad \forall i \in I^v, \quad \forall a \in A^r, \quad \forall p = (r_1, r_2) \in P_i^v \quad (4.30)$$

Equation (4.31), in turn, assures that, for a pair of racks exchanging data, the amount of traffic leaving the outgoing rack will be the same that the one entering the incoming rack.

$$\sum_{w_1 \in S_i^v | (w_1,r_2) \in E_i^v} F_{a,i,w_1,r_2,p} = \sum_{w_2 \in S_i^v | (r_1,w_2) \in E_i^v} F_{a,i,r_1,w_2,p} \quad \forall i \in I^v, \quad \forall a \in A^r, \quad \forall p = (r_1, r_2) \in P_i^v \quad (4.31)$$

Equation (4.32) guarantees flow conservation at intermediary switches of the path and Equations (4.33), (4.34) and (4.35) ensure that each flow has only one path in the network (by setting variable $h_{a,i,(w_1,w_2),p}$ only if link (w_1, w_2) is used by $p = (r_1, r_2) \in P_i^v$ for communication of VMs from application a).

$$\sum_{\substack{w_4 \in S_i^v \setminus \{r_1, w_3\} | \\ (w_3,w_4) \in E_i^v}} F_{a,i,(w_3,w_4),p} - \sum_{\substack{w_4 \in S_i^v \setminus \{r_2, w_3\} | \\ (w_4,w_3) \in E_i^v}} F_{a,i,(w_4,w_3),p} = 0$$
$$\forall i \in I^v, \quad \forall a \in A^r, \quad \forall p = (r_1, r_2) \in P_i^v \quad \forall w_3 \in S_i^v \setminus \{r_1, r_2\} \quad (4.32)$$

$$F_{a,i,(w_1,w_2),p} \leq Band^v(w_1, w_2) * h_{a,i,(w_1,w_2),p}$$
$$\forall a \in A^r, \quad \forall i \in I^v, \quad \forall w_1, w_2 \in S_i^v \mid (w_1, w_2) \in E_i^v, \quad \forall p \in P_i^v \quad (4.33)$$

$$\sum_{\substack{w_2 \in S_i^v | \\ (w_1,w_2) \in E_i^v}} h_{a,i,(w_1,w_2),p} \leq 1 \qquad \forall a \in A^r, \quad \forall i \in I^v, \quad \forall w_1 \in S_i^v, \quad \forall p \in P_i^v \quad (4.34)$$

$$h_{a,i,(w_1,w_2),(r_1,r_2)} + h_{a,i,(w_1,w_2),(r_2,r_1)} \leq 1 \qquad \forall a \in A^r, \quad \forall i \in I^v, \quad \forall p = (r_1, r_2) \in P_i^v,$$
$$\forall w_1 \in R_i^v, \quad \forall w_2 \in S_i^v \mid w_1 \neq w_2 \text{ and } (w_1, w_2) \in E_i^v \quad (4.35)$$

The last two constraint sets, Equations (4.36) and (4.37), ensure flow conservation between switches ToR and VMs. Specifically, Equation (4.36) allocates the requested bandwidth for the link between a VM and its ToR switch, while Equations (4.37) guarantees that all links are symmetric.

$$\sum_{p \in P_i^v} F_{a,i,(w_1,w_2),p} = g_{a,i,w_2,w_1} * Band_a^r \quad \forall a \in A^r,$$
$$\forall i \in I^v, \quad \forall w_1 \in M_i^v, \quad \forall w_2 \in R_i^v \mid (w_1, w_2) \in E_i^v \quad (4.36)$$

$$\sum_{p \in P_i^v} F_{a,i,(w_1,w_2),p} = \sum_{p \in P_i^v} F_{a,i,(w_2,w_1),p} \qquad \forall i \in I^v,$$
$$\forall a \in A^r, \quad \forall w_1 \in M_i^v, \quad \forall w_2 \in R_i^v \mid (w_1, w_2) \in E_i^v \quad (4.37)$$

4.3 Optimal Resource Allocation Evaluation

In this section, we focus on evaluating two aspects of the proposed strategy: *i*) the solution quality, which is measured considering the benefit provided by grouping

applications into VIs (i.e., the increased level of security); and *ii*) the cost (resource fragmentation), due to the use of VIs. We also verify the processing time required for the optimal resource allocation.

Evaluation Environment. The mathematical formulation described in this chapter was implemented on IBM ILOG CPLEX Optimization Studio 12.3[4]. CPLEX uses variants of the Simplex algorithm and Branch-and-Bound technique to find optimal solutions within an error margin (i.e., "gap to optimality"). All experiments were performed in an Intel Core i3-2120 running at 3.30 GHz, equipped with 8 GB of RAM and operating system GNU/Linux Debian x86_64. Each experiment is limited to a maximum duration of 1 hour and with a search tree of at most 4 GB, due to the problem complexity. Thus, the results shown are the best feasible ones achieved by CPLEX during the period.

Cloud datacenter infrastructure. Similarly to related work (BALLANI et al., 2011b; GUO et al., 2010a; SHIEH et al., 2011; XIE et al., 2012), the physical substrate (cloud datacenter infrastructure) was defined as a tree-like topology. The substrate is composed of 60 physical servers, each one with 4 available slots to host VMs, equally divided into 10 racks. Racks are connected by switches at upper layers with path diversity.

Workload. The workload is composed of batches of 10, 13 and 16 requests to instantiate applications in the cloud, each one from a distinct tenant. The number of VMs and bandwidth specified in each request is uniformly distributed within [2, 5] and [50, 300] Mbps, respectively. Mutual trust between tenants was generated by means of 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 no path diversity and similar size in comparison to the other VIs from the set.

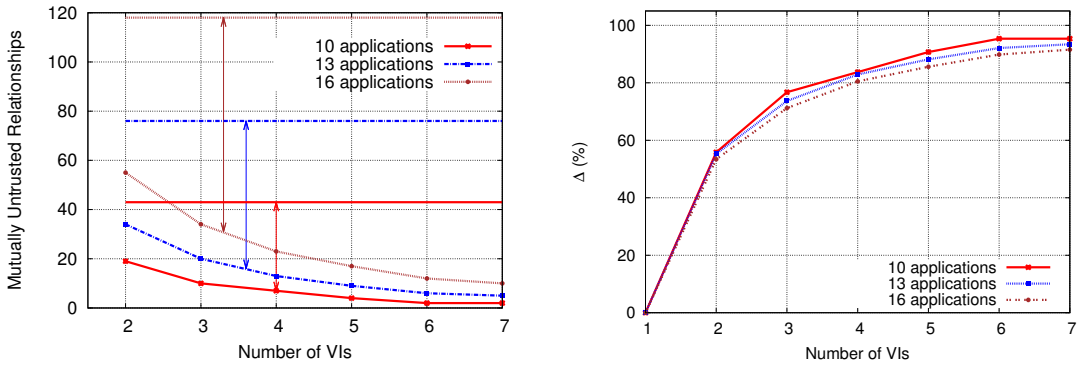
The use of such small values for the physical substrate and workload was necessary to perform the set of experiments, since CPLEX consumes a lot of CPU and memory resources. This limitation is discussed at the end of this section.

Solution Quality. The solution quality is measured considering the benefit provided by grouping applications into VIs. That is, we seek to evaluate the solution based on the security offered by grouping tenant applications in comparison with the baseline (current) scenario, in which there is no grouping (i.e., all tenants share the same network). Thus, the metric employed is the number of mutually untrusted relationships between tenant applications that were assigned to the same VI. It is desirable to have this value minimized, because it may expose applications to attacks.

Figure 4.3(a) shows the variation of the number of mutually untrusted relationships in accordance with the number of VIs offered by the provider for different batches of application requests. We then define the trust level mean difference when applications are grouped together: $\Delta = 1 - \frac{\Phi}{\Gamma}$, where Φ is the number of untrusted relationships between applications within each group and Γ denotes the number of mutually untrusted relationships without grouping. Δ is shown in Figure 4.3(b),

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

which demonstrates that the number of applications is not the main factor to increase security, but the number of VIs offered by the provider. In general, we see that security increases with a logarithmic behavior according to the number of VIs.



(a) Untrusted relationships. (b) Security level difference.

Figure 4.3: Security provided by grouping applications.

The grouping of applications can also be beneficial to reduce the bandwidth consumed and to achieve isolation among applications with distinct bandwidth requirements. VMs that belong to the same application are positioned, by definition, within a VI, while resources from each VI are likely to be allocated close to each other on the physical infrastructure. Therefore, the smaller the VI, the closer the communicating VMs and, hence, the lesser the amount of network resources consumed. The isolation among applications with different bandwidth requirements, in turn, minimizes the harmful interference that can be caused by mixing applications with distinct traffic characteristics (ABTS; FELDERMAN, 2012b).

On the other hand, the creation of VIs and the grouping of tenants introduces two potentially negative effects. First, the creation and maintenance of VIs introduce management and communication overheads in comparison with a conventional network. Nonetheless, the overhead is small and fully justifiable considering the benefits for both tenants and providers (BALLANI et al., 2011b). Second, the grouping of applications may restrain the allocation of cloud requests, because of (internal) resource fragmentation. This may increase the number of rejected requests, even if the sum of available resources considering all VIs would be enough to allocate the application.

To this end, Figure 4.4 shows the overall acceptance ratio of requests during 70 time units. The arrival rate of each request is given by a Poisson distribution with an average of 5 requests per time unit. At first, all application requests are accepted because the cloud has ample resources. As time passes and the load increases, requests start to be dropped, resulting in a convergence period. Since rejections occur at different loads for different configurations, we can see a brief moment where the best alternative is unclear. Finally, the allocation converges after time 35. In particular, we see that the acceptance ratio stabilizes around 0.6, showing negligible overhead (acceptance 1.5% lower) than the baseline scenario. Thus, it is possible to increase security with a minimum addition to the rejection rate in comparison to the baseline scenario.

Allocation cost. Function $\mathcal{G}$ complexity is combinatorial in accordance with

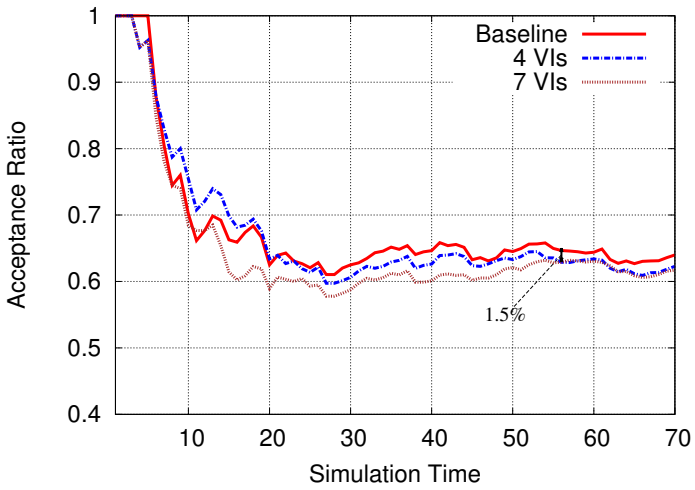


Figure 4.4: Overall acceptance ratio of requests.

the set of VIs and the physical substrate. Therefore, CPLEX may take many hours to optimally solve Function $\mathcal{G}$. The time taken, however, is highly variable, since it depends on the location of the optimal solution on the solution tree generated by CPLEX (and the tree depends on the input data). Nonetheless, the time taken may still be acceptable, since Function $\mathcal{G}$ needs to be executed only when VIs are allocated on the cloud substrate. Should the time taken be high, virtual network embedding heuristics, such as D-ViNE and R-ViNE (CHOWDHURY; RAHMAN; BOUTABA, 2009), can be adapted to perform this operation. Function $\mathcal{F}$ complexity, in turn, is combinatorial according to the number of applications, the number of VIs and their respective virtual elements. In this sense, the solver may take several minutes to find a feasible (or the optimal) solution for Function $\mathcal{F}$ (specially in large-scale cloud platforms). Given the environment dynamism (i.e., churn) and that Function $\mathcal{F}$ must be executed to allocate every incoming application, the time taken is not suitable. Therefore, application allocation in large-scale cloud datacenters should be carried on by a constructive heuristic.

5 EMBEDDING HEURISTIC

The functions $\mathcal{F}$ and $\mathcal{G}$, defined in the previous chapter, cannot be optimally solved in reasonable time for the application in mind. Therefore, heuristics for functions $\mathcal{F}$ and $\mathcal{G}$ are needed. One can find heuristics for $\mathcal{G}$ in the literature, such as Yu et al. (2008), Chowdhury, Rahman & Boutaba (2009) and Zhang et al. (2012). The same does not apply for $\mathcal{F}$. This chapter covers this gap by proposing a heuristic for function $\mathcal{F}$.

Function $\mathcal{F}$ 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. Despite this, the high rate of tenant arrival and departure requires the operation to be performed as quickly as possible. In general, it is computationally expensive to employ optimization strategies (YU et al., 2008), such as iterative methods (LU; TURNER, 2006) and simulated annealing (FAN; AMMAR, 2006; SKIŚCIM; GOLDEN, 1983). 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, attempt to minimize bandwidth consumption for intra-application communication.

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 increasing the likelyhood of allocating all VMs from the application close to each other in order to address network performance as well.

The algorithm initializes the set of VM clusters C_a^r 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 the ToR switch 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, function `DeallocatePartiallyAllocatedRequest` deallocates the set of resources that could have been reserved for application a during the iteration in VI i. The algorithm, then, 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.

```
Input : Application a, Virtual infrastructure set I^v
unvisitedVIs ← GetVIs(I^v);
while true do
    i ← SelectVI(unvisitedVIs, T^r_{a,x});
    if not i then return false;
    unvisitedVIs ← unvisitedVIs \ {i};
    C^r_a ← ∅;
    r^v ← FindBestRack(i, M^r_a);
    maxVMs ← MaxAvailableCluster(r^v, M^r_a, Band^r_a);
    C^r_a ← C^r_a ∪ {Cluster(r^v, maxVMs)};
    allocatedVMs ← maxVMs;
    if allocatedVMs < M^r_a then
        switchQueue ← FindNeighborSwitches(r^v);
        while allocatedVMs < M^r_a do
            s^v ← GetSwitch(switchQueue);
            if not s^v then break;
            switchQueue ← FindNeighborSwitches(s^v);
            torQueue ← FindRacks(s^v);
            while torQueue not empty do
                r^v ← GetToR(torQueue);
                maxVMs ← MaxAvailableCluster(r^v, (M^r_a − allocatedVMs),
                Band^r_a);
                C^r_a ← C^r_a ∪ Cluster(r^v, maxVMs);
                allocatedVMs ← allocatedVMs + maxVMs;
                if allocatedVMs == M^r_a then break;
            end
        end
    end
    if allocatedVMs == M^r_a and AllocBandwidth(C^r_a) then
        return true;
    end
    else
        DeallocatePartiallyAllocatedRequest(a);
    end
end
```

Algorithm 1: Application allocation algorithm.

6 EVALUATION

In this chapter, 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.

6.1 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 and implements our constructive heuristic. We focus on tree-like topologies such as multi-rooted trees used in today's datacenters (SHIEH et al., 2011). 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 Xie et al. (2012) and Ballani et al. (2011b).

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 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.

6.2 Evaluation Results

Improved security and performance for tenants. Similarly to the optimal allocation evaluation presented in Section 4.3, 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 shows how exposed applications are 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 6.1 depicts the variation of mutually untrusted relationships for three batches of application requests in accordance with the number of VIs offered by the provider. We confirm that the security increases with a logarithmic behavior according to the number of VIs for large-scale cloud platforms as well (similarly to the optimal allocation results with a reduced set of resources). Therefore, our strategy can scale and improve security for tenants.

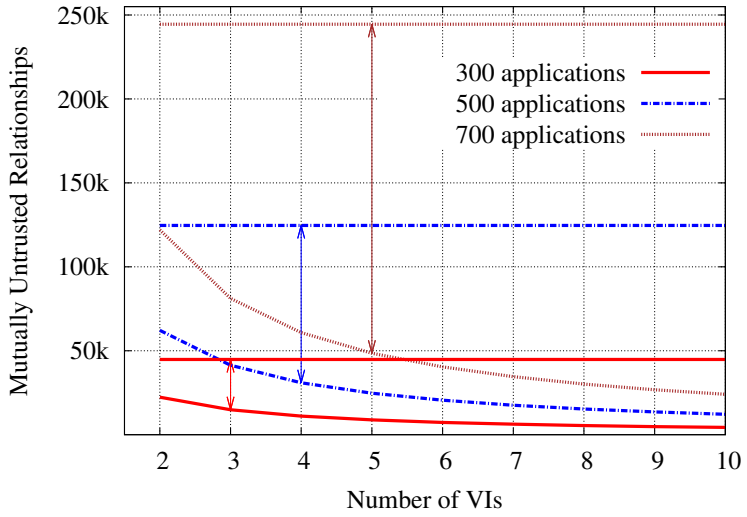


Figure 6.1: Security when allocating batches of applications.

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 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 6.2 shows how the number of 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 (around time 200), 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, opportunities for 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.

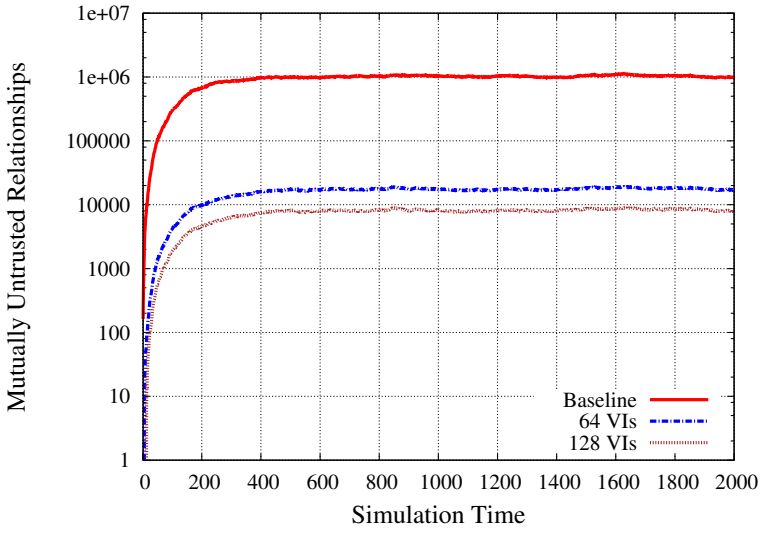
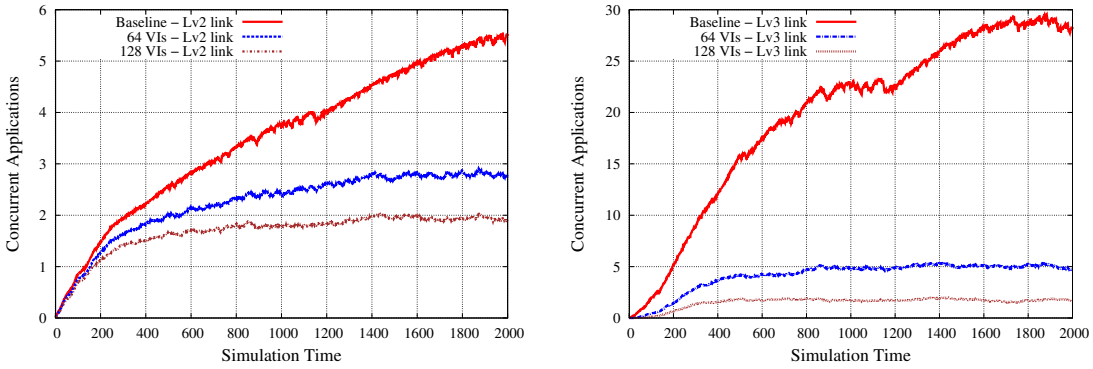


Figure 6.2: Security in an online setting.

We also verify the number of applications competing for bandwidth in level-2 and level-3 links of the virtual tree topologies. Figure 6.3(a) 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 6.3(b) 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 interference in the network is one of the leading causes for poor application performance in the cloud (SHIEH et al., 2011; BALLANI et al., 2011b). 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.

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 6.4 shows results regarding internal fragmentation of resources. Figure 6.4(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



(a) Level-2 links.

(b) Level-3 links.

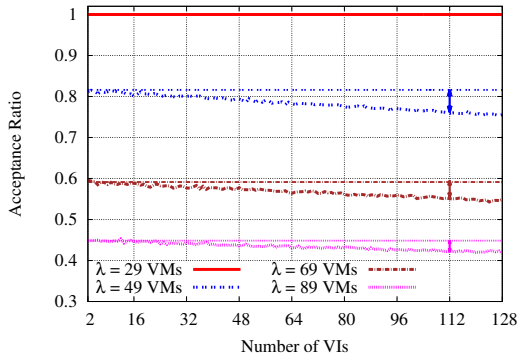
Figure 6.3: Mean number of applications sharing links at distinct levels.

(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 value of λ and, on a smaller scale, the excessive use of virtual infrastructures may lead to resource fragmentation inside the cloud infrastructure.

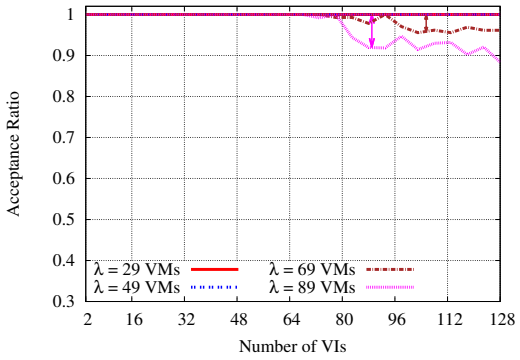
Figure 6.4(b) depicts the acceptance ratio with cloud-load between 70% and 80% (i.e., the usual load of public cloud platforms, such as Amazon EC2 (BIAS, 2011)). We see that the acceptance ratio decreases according to the number of VIs offered and the size of applications (that is, the bigger the applications 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 6.4(c), in turn, shows the acceptance ratio of requests for $\lambda = 49$ (i.e., a setting closer to reality (SHIEH et al., 2011)) 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). Finally, Figure 6.4(d) shows the cloud-load during a predefined time period. At first, the load is low and increases according to the allocation of incoming application requests. As time passes, it tends to stabilize around 92% for 128 VIs, 96% for 64 VIs and 99% for the baseline scenario.

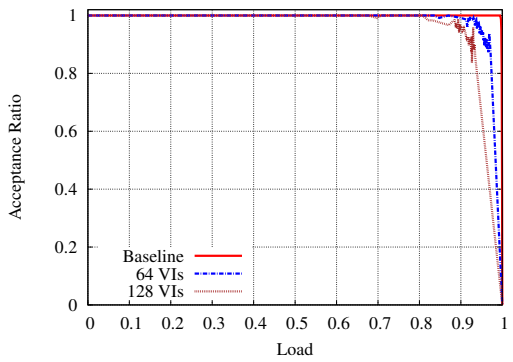
Although resource fragmentation is a side-effect of our approach, we consider that this burden is pragmatically negligible for realistic datacenter loads (BIAS, 2011) (and results, in particular from Figures 6.4(b) and 6.4(c), show that, at this load, fragmentation is small). Furthermore, our strategy 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



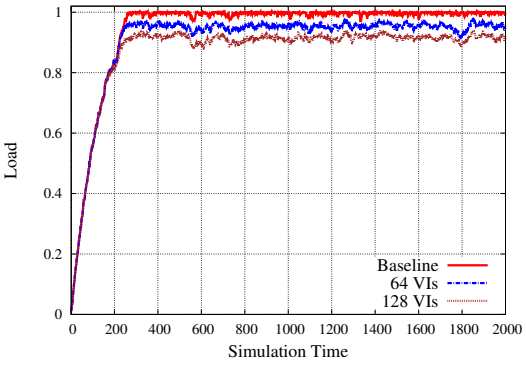
(a) Overall acceptance ratio according to the number of VIs.



(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).



(d) Cloud-load during a predefined time period ($\lambda = 49$ VMs).

Figure 6.4: Internal resource fragmentation.

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 developing fair and efficient pricing models is still ongoing research (BALLANI et al., 2011a; NIU; FENG; LI, 2012), we adopt a simple pricing model similar to Ballani et al. (2011b) and Xie et al. (2012), 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 6.5 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.

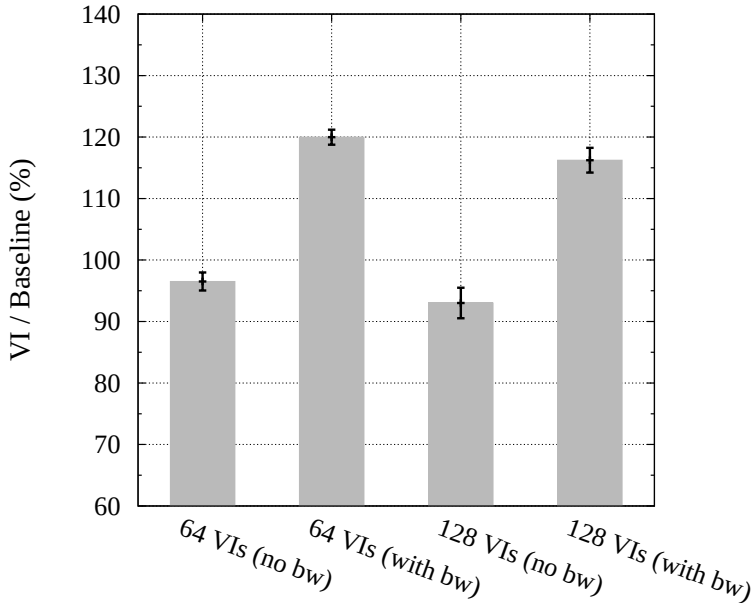


Figure 6.5: Provider revenue.

7 CONCLUSIONS AND FUTURE WORK

Cloud Computing has become the platform of choice for the delivery and consumption of IT resources through a pay-per-use pricing model (that is, tenants are charged by the amount of resources and the time consumed to execute applications). In this model, providers seek to achieve economies of scale by implementing datacenters as highly multiplexed shared environments, where all tenants share the same set of physical resources.

However, the lack of mechanisms to control network resource sharing inside the cloud enables selfish and malicious use of the network. Selfish tenants consume an unfair share of the network (and, consequently, cast performance interference attacks) by using non-compliant versions of TCP, while malicious ones launch DoS and other types of attacks (such as man-in-the-middle and extraction of confidential information). Such attacks hurt both tenants and providers. Thus, it is essential to take security and performance into account when allocating applications in cloud platforms.

In this thesis, we have proposed 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. Our strategy is composed of two steps. The first one allocates VIs on the physical substrate, while the second one distributes and maps applications into VIs according to the mutual trust relationships between tenants. Thus, the environment becomes more resilient against adversaries that could hurt 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.

Our key contribution is the design of a cloud resource allocation scheme that increases the security and performance of applications (what tenants desire) with low impact on cloud resource utilization (what providers desire). The remaining contributions are twofold: *i*) we formally present the proposed strategy as a MIP optimization model and propose a constructive heuristic to efficiently allocate tenant applications in large-scale cloud platforms; *ii*) we evaluate the trade-off between the gain in security and performance for tenants and the cost imposed on cloud

providers by our solution, and show that the proposed approach can substantially increase security and performance with low resource fragmentation.

In future work, we consider improving the strategy in four ways. First, we plan to include security requirements and virtual link embedding into multiple paths in function $\mathcal{G}$ (i.e., when mapping VIs onto the physical substrate). Therefore, we can increase isolation among VIs in the physical substrate and resiliency in case of physical resource failure. Second, we aim at developing new algorithms to provide horizontal elasticity (the ability to add or remove VMs from an already allocated application), thus supporting elastic growth of applications during their lifetime. Third, we intend to dynamically reduce or increase virtual infrastructure size in order to improve security and minimize resource fragmentation. Fourth, we plan on developing metaheuristics to optimize the use of cloud resources as time passes by. This requires the migration of already allocated VMs, which must be performed carefully, since we need to consider the benefits and losses for both providers and tenants.

REFERENCES

ABTS, D.; FELDERMAN, B. A Guided Tour through Data-center Networking. **Queue**, ACM, New York, NY, USA, v. 10, n. 5, p. 10:10–10:23, maio 2012. ISSN 1542-7730.

ABTS, D.; FELDERMAN, B. A guided tour of data-center networking. **Commun. ACM**, ACM, New York, NY, USA, v. 55, n. 6, p. 44–51, jun. 2012. ISSN 0001-0782.

ABU-LIBDEH, H.; COSTA, P.; ROWSTRON, A.; O'SHEA, G.; DONNELLY, A. Symbiotic routing in future data centers. In: ACM SIGCOMM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION, 38., 2010, SIGCOMM, New Delhi, IN. **Proceedings...** New York, NY, USA: ACM, 2010. P. 51–62. ISBN 978-1-4503-0201-2.

AL-FARES, M.; LOUKISSAS, A.; VAHDAT, A. A scalable, commodity data center network architecture. In: ACM SIGCOMM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION, 36., 2008, SIGCOMM, Seattle, WA, USA. **Proceedings...** New York, NY, USA: ACM, 2008. P. 63–74. ISBN 978-1-60558-175-0.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R. H.; KONWINSKI, A.; LEE, G.; PATTERSON, D. A.; RABKIN, A.; STOICA, I.; ZAHARIA, M. **Above the Clouds: A Berkeley View of Cloud Computing**. Berkeley, CA, USA, fev. 2009. Available at: `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html`. Visited on: Apr. 20, 2011.

ARMBRUST, M.; FOX, A.; GRIFFITH, R.; JOSEPH, A. D.; KATZ, R.; KONWINSKI, A.; LEE, G.; PATTERSON, D.; RABKIN, A.; STOICA, I.; ZAHARIA, M. A view of cloud computing. **Commun. ACM**, ACM, New York, NY, USA, v. 53, n. 4, p. 50–58, abr. 2010. ISSN 0001-0782.

BALLANI, H.; COSTA, P.; KARAGIANNIS, T.; ROWSTRON, A. The price is right: towards location-independent costs in datacenters. In: WORKSHOP ON HOT TOPICS IN NETWORKS, 10., 2011, HotNets, Cambridge, MA, USA. **Proceedings...** New York, NY, USA: ACM, 2011. P. 23:1–23:6. ISBN 978-1-4503-1059-8.

BALLANI, H.; COSTA, P.; KARAGIANNIS, T.; ROWSTRON, A. Towards predictable datacenter networks. In: ACM SIGCOMM CONFERENCE ON

APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION, 39., 2011, SIGCOMM, Toronto, ON, Canada. **Proceedings...** New York, NY, USA: ACM, 2011. P. 242–253. ISBN 978-1-4503-0797-0.

BARHAM, P.; DRAGOVIC, B.; FRASER, K.; HAND, S.; HARRIS, T.; HO, A.; NEUGEBAUER, R.; PRATT, I.; WARFIELD, A. Xen and the art of virtualization. In: ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, 19., 2003, SOSP, Bolton Landing, NY USA. **Proceedings...** New York, NY, USA: ACM, 2003. P. 164–177. ISBN 1-58113-757-5.

BARI, M.; BOUTABA, R.; ESTEVES, R.; GRANVILLE, L.; PODLESNY, M.; RABBANI, M.; ZHANG, Q.; ZHANI, F. Data Center Network Virtualization: A Survey. **IEEE Comm. Surv. Tut.**, PP, n. 99, 2012.

BENSON, T.; AKELLA, A.; MALTZ, D. A. Network traffic characteristics of data centers in the wild. In: INTERNET MEASUREMENT CONFERENCE, 10., 2010, IMC, Melbourne, AU. **Proceedings...** New York, NY, USA: ACM, 2010. P. 267–280. ISBN 978-1-4503-0483-2.

BENSON, T.; AKELLA, A.; SHAIKH, A.; SAHU, S. CloudNaaS: a cloud networking platform for enterprise applications. In: ACM SYMPOSIUM ON CLOUD COMPUTING, 2., 2011, SoCC, Cascais, PT. **Proceedings...** New York, NY, USA: ACM, 2011. P. 8:1–8:13. ISBN 978-1-4503-0976-9.

BIAS, R. **Amazon's EC2 Generating 220M**. 2011. Available at: `http://cloudscaling.com/blog/cloud-computing/amazons-ec2-generating-220m-annually`. Visited on: Mar. 20, 2012.

BREITGAND, D.; EPSTEIN, A. Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS, 31., 2012, INFOCOM, Orlando, FL, USA. **Proceedings...** Piscataway, NJ, USA: IEEE Press, 2012. P. 2861–2865. ISSN 0743-166X.

CHOWDHURY, N.; RAHMAN, M.; BOUTABA, R. Virtual network embedding with coordinated node and link mapping. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS, 28., 2009, INFOCOM, Rio de Janeiro, RJ, Brazil. **Proceedings...** Piscataway, NJ, USA: IEEE Press, 2009. P. 783–791. ISSN 0743-166X.

CURTIS, A.; CARPENTER, T.; ELSHEIKH, M.; LOPEZ-ORTIZ, A.; KESHAV, S. REWIRE: An optimization-based framework for unstructured data center network design. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS, 31., 2012, INFOCOM, Orlando, FL, USA. **Proceedings...** Piscataway, NJ, USA: IEEE Press, 2012. P. 1116–1124. ISSN 0743-166X.

CURTIS, A. R.; KESHAV, S.; LOPEZ-ORTIZ, A. LEGUP: using heterogeneity to reduce the cost of data center network upgrades. In: INTERNATIONAL CONFERENCE ON EMERGING NETWORKING EXPERIMENTS AND TECHNOLOGIES, 6., 2010, CoNEXT, Philadelphia, PA, USA. **Proceedings...** New York, NY, USA: ACM, 2010. P. 14:1–14:12. ISBN 978-1-4503-0448-1.

DALLY, W.; TOWLES, B. **Principles and Practices of Interconnection Networks**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003. ISBN 0122007514.

FAN, J.; AMMAR, M. H. Dynamic Topology Configuration in Service Overlay Networks: A Study of Reconfiguration Policies. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS, 25., 2006, INFOCOM, Barcelona, ES. **Proceedings...** Piscataway, NJ, USA: IEEE Press, 2006. P. 1–12. ISSN 0743-166X.

FARRINGTON, N.; RUBOW, E.; VAHDAT, A. Data Center Switch Architecture in the Age of Merchant Silicon. In: IEEE SYMPOSIUM ON HIGH PERFORMANCE INTERCONNECTS, 17., 2009, HOTI, New York, NY, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2009. P. 93–102. ISSN 1550-4794.

FLOYD, S.; HANDLEY, M.; PADHYE, J.; WIDMER, J. **TCP Friendly Rate Control (TFRC): Protocol Specification**. 2008.

GIURGIU, A. **Network performance in virtual infrastrucures**. February 2010. Available at: `http://staff.science.uva.nl/~delaat/sne-2009-2010/p29/presentation.pdf`. Visited on: Jan. 20, 2012.

GOEL, A.; INDYK, P. Stochastic load balancing and related problems. In: IEEE SYMPOSIUM ON FOUNDATIONS OF COMPUTER SCIENCE, 40., 1999, FOCS, New York, NY, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 1999. P. 579 –586. ISSN 0272-5428.

GREENBERG, A.; HAMILTON, J.; MALTZ, D. A.; PATEL, P. The cost of a cloud: research problems in data center networks. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 39, n. 1, p. 68–73, dez. 2008. ISSN 0146-4833.

GREENBERG, A.; HAMILTON, J. R.; JAIN, N.; KANDULA, S.; KIM, C.; LAHIRI, P.; MALTZ, D. A.; PATEL, P.; SENGUPTA, S. VL2: a scalable and flexible data center network. In: ACM SIGCOMM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION, 37., 2009, SIGCOMM, Barcelona, ES. **Proceedings...** New York, NY, USA: ACM, 2009. P. 51–62. ISBN 978-1-60558-594-9.

GREENBERG, A.; HAMILTON, J. R.; JAIN, N.; KANDULA, S.; KIM, C.; LAHIRI, P.; MALTZ, D. A.; PATEL, P.; SENGUPTA, S. Vl2: a scalable and flexible data center network. **Commun. ACM**, ACM, New York, NY, USA, v. 54, n. 3, p. 95–104, mar. 2011. ISSN 0001-0782.

GUO, C.; LU, G.; LI, D.; WU, H.; ZHANG, X.; SHI, Y.; TIAN, C.; ZHANG, Y.; LU, S. Bcube: a high performance, server-centric network architecture for modular data centers. In: ACM SIGCOMM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION, 37., 2009, SIGCOMM, Barcelona, ES. **Proceedings...** New York, NY, USA: ACM, 2009. P. 63–74. ISBN 978-1-60558-594-9.

GUO, C.; LU, G.; WANG, H. J.; YANG, S.; KONG, C.; SUN, P.; WU, W.; ZHANG, Y. SecondNet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees. In: INTERNATIONAL CONFERENCE ON EMERGING NETWORKING EXPERIMENTS AND TECHNOLOGIES, 6., 2010, CoNEXT, Philadelphia, PA, USA. **Proceedings...** New York, NY, USA: ACM, 2010. P. 15:1–15:12. ISBN 978-1-4503-0448-1.

GUO, C.; LU, G.; WANG, H. J.; YANG, S.; KONG, C.; SUN, P.; WU, W.; ZHANG, Y. **SecondNet: A Data Center Network Virtualization Architecture with Bandwidth Guarantees**. Haidian District, Beijing, China, 2010.

GUO, C.; WU, H.; TAN, K.; SHI, L.; ZHANG, Y.; LU, S. Dcell: a scalable and fault-tolerant network structure for data centers. In: ACM SIGCOMM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION, 36., 2008, SIGCOMM, Seattle, WA, USA. **Proceedings...** New York, NY, USA: ACM, 2008. P. 75–86. ISBN 978-1-60558-175-0.

IEEE COMPUTER SOCIETY. **IEEE std. 802.1Q, Virtual Bridged Local Area Networks**. 2005.

IOSUP, A.; YIGITBASI, N.; EPEMA, D. On the performance variability of production cloud services. In: IEEE/ACM INTERNATIONAL SYMPOSIUM ON CLUSTER, CLOUD AND GRID COMPUTING, 11., 2011, CCGRID, Newport Beach, CA, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2011. P. 104–113. ISBN 978-0-7695-4395-6.

JENSEN, M.; SCHWENK, J.; GRUSCHKA, N.; IACONO, L. L. On technical security issues in cloud computing. In: IEEE INTERNATIONAL CONFERENCE ON CLOUD COMPUTING, 1., 2009, CLOUD, Bangalore, IN. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2009. P. 109–116. ISBN 978-0-7695-3840-2.

KITSOS, I.; PAPAIOANNOU, A.; TSIKOUDIS, N.; MAGOUTIS, K. Adapting data-intensive workloads to generic allocation policies in cloud infrastructures. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, 13., 2012, NOMS, Maui, HI, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2012. P. 25–33. ISSN 1542-1201.

KLEINBERG, J.; RABANI, Y.; TARDOS, E. Allocating bandwidth for bursty connections. In: ACM SYMPOSIUM ON THEORY OF COMPUTING, 29., 1997, STOC, El Paso, TX, USA. **Proceedings...** New York, NY, USA: ACM, 1997. P. 664–673. ISBN 0-89791-888-6.

KOHLER, E.; HANDLEY, M.; FLOYD, S. **Datagram Congestion Control Protocol (DCCP)**. 2006.

KOSSMANN, D.; KRASKA, T.; LOESING, S. An evaluation of alternative architectures for transaction processing in the cloud. In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 36., 2010, SIGMOD,

Indianapolis, IN, USA. **Proceedings...** New York, NY, USA: ACM, 2010. P. 579–590. ISBN 978-1-4503-0032-2.

LAM, V. T.; RADHAKRISHNAN, S.; PAN, R.; VAHDAT, A.; VARGHESE, G. Netshare and stochastic netshare: predictable bandwidth allocation for data centers. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 42, n. 3, p. 5–11, 2012. ISSN 0146-4833.

LI, A.; YANG, X.; KANDULA, S.; ZHANG, M. Cloudcmp: comparing public cloud providers. In: INTERNET MEASUREMENT CONFERENCE, 10., 2010, IMC, Melbourne, AU. **Proceedings...** New York, NY, USA: ACM, 2010. P. 1–14. ISBN 978-1-4503-0483-2.

LIU, H. A new form of DOS attack in a cloud and its avoidance mechanism. In: ACM CLOUD COMPUTING SECURITY WORKSHOP, 2., 2010, CCSW, Chicago, IL, USA. **Proceedings...** New York, NY, USA: ACM, 2010. P. 65–76.

LU, J.; TURNER, J. **Efficient Mapping of Virtual Networks onto a Shared Substrate**. St. Louis, Missouri, USA, 2006. Available at: `http://www.arl.wustl.edu/~jl1/research/tech\_report\_2006.pdf`. Visited on: Feb 5, 2012.

MANGOT, D. **Measuring EC2 system performance**. May 2009. Available at: `http://bit.ly/48Wui`. Visited on: Mar. 12, 2012.

MENG, X.; PAPPAS, V.; ZHANG, L. Improving the scalability of data center networks with traffic-aware virtual machine placement. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS, 29., 2010, INFOCOM, San Diego, CA, USA. **Proceedings...** Piscataway, NJ, USA: IEEE Press, 2010. P. 1154–1162. ISBN 978-1-4244-5836-3.

MUDIGONDA, J.; YALAGANDULA, P.; AL-FARES, M.; MOGUL, J. C. SPAIN: COTS data-center Ethernet for multipathing over arbitrary topologies. In: USENIX CONFERENCE ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION, 7., 2010, NSDI, San Jose, CA, USA. **Proceedings...** Berkeley, CA, USA: USENIX Association, 2010. P. 18–18.

MUDIGONDA, J.; YALAGANDULA, P.; MOGUL, J.; STIEKES, B.; POUFFARY, Y. Netlord: a scalable multi-tenant network architecture for virtualized datacenters. In: ACM SIGCOMM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION, 39., 2011, SIGCOMM, Toronto, ON, Canada. **Proceedings...** New York, NY, USA: ACM, 2011. P. 62–73. ISBN 978-1-4503-0797-0.

NIU, D.; FENG, C.; LI, B. Pricing cloud bandwidth reservations under demand uncertainty. In: INTERNATIONAL CONFERENCE ON MEASUREMENT AND MODELING OF COMPUTER SYSTEMS, 12., 2012, SIGMETRICS/PERFORMANCE, London, UK. **Proceedings...** New York, NY, USA: ACM, 2012. P. 151–162. ISBN 978-1-4503-1097-0.

POPA, L.; KUMAR, G.; CHOWDHURY, M.; KRISHNAMURTHY, A.; RATNASAMY, S.; STOICA, I. FairCloud: sharing the network in cloud computing. In: ACM SIGCOMM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION, 40., 2012, SIGCOMM, Helsinki, FI. **Proceedings...** New York, NY, USA: ACM, 2012. P. 187–198. ISBN 978-1-4503-1419-0.

PRAKASH, P.; DIXIT, A.; HU, Y. C.; KOMPELLA, R. The TCP outcast problem: exposing unfairness in data center networks. In: USENIX CONFERENCE ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION, 9., 2012, NSDI, San Jose, CA, USA. **Proceedings...** Berkeley, CA, USA: USENIX Association, 2012. P. 30–30.

RISTENPART, T.; TROMER, E.; SHACHAM, H.; SAVAGE, S. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: ACM CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY, 16., 2009, CCS, Chicago, IL, USA. **Proceedings...** New York, NY, USA: ACM, 2009. P. 199–212.

RODRIGUES, H.; SANTOS, J. R.; TURNER, Y.; SOARES, P.; GUEDES, D. Gatekeeper: supporting bandwidth guarantees for multi-tenant datacenter networks. In: WORKSHOP ON I/O VIRTUALIZATION, 3., 2011, WIOV, PORTLAND, OR, USA. **Proceedings...** Berkeley, CA, USA: USENIX Association, 2011. P. 6–6.

SCHAD, J.; DITTRICH, J.; QUIANÉ-RUIZ, J.-A. Runtime measurements in the cloud: observing, analyzing, and reducing variance. **VLDB Endowment**, v. 3, n. 1-2, p. 460–471, Sept. 2010. ISSN 2150-8097.

SHIEH, A.; KANDULA, S.; GREENBERG, A.; KIM, C. Seawall: performance isolation for cloud datacenter networks. In: USENIX WORKSHOP ON HOT TOPICS IN CLOUD COMPUTING, 2., 2010, HotCloud, Boston, MA, USA. **Proceedings...** Berkeley, CA, USA: USENIX Association, 2010. P. 1–1.

SHIEH, A.; KANDULA, S.; GREENBERG, A.; KIM, C.; SAHA, B. Sharing the data center network. In: USENIX CONFERENCE ON NETWORKED SYSTEMS DESIGN AND IMPLEMENTATION, 8., 2011, NSDI, Boston, MA, USA. **Proceedings...** Berkeley, CA, USA: USENIX Association, 2011. P. 23–23.

SKIŚCIM, C. C.; GOLDEN, B. L. Optimization by simulated annealing: A preliminary computational study for the TSP. In: WINTER SIMULATION CONFERENCE, 15., 1983, WSC, Arlington, VA, USA. **Proceedings...** Piscataway, NJ, USA: IEEE Press, 1983. P. 523–535.

WANG, G.; NG, T. S. E. The impact of virtualization on network performance of amazon ec2 data center. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS, 29., 2010, INFOCOM, San Diego, CA, USA. **Proceedings...** Piscataway, NJ, USA: IEEE Press, 2010. P. 1163–1171. ISBN 978-1-4244-5836-3.

WANG, M.; MENG, X.; ZHANG, L. Consolidating virtual machines with dynamic bandwidth demand in data centers. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS, 30., 2011, INFOCOM, Shanghai, CN. **Proceedings...** Piscataway, NJ, USA: IEEE Press, 2011. P. 71 –75. ISSN 0743-166X.

XIE, D.; DING, N.; HU, Y. C.; KOMPELLA, R. The Only Constant is Change: Incorporating Time-Varying Network Reservations in Data Centers. In: ACM SIGCOMM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION, 40., 2012, SIGCOMM, Helsinki, FI. **Proceedings...** New York, NY, USA: ACM, 2012. P. 199–210. ISBN 978-1-4503-1419-0.

YU, M.; YI, Y.; REXFORD, J.; CHIANG, M. Rethinking virtual network embedding: substrate support for path splitting and migration. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 38, p. 17–29, March 2008. ISSN 0146-4833.

ZAHARIA, M.; KONWINSKI, A.; JOSEPH, A. D.; KATZ, R.; STOICA, I. Improving MapReduce performance in heterogeneous environments. In: USENIX SYMPOSIUM ON OPERATING SYSTEMS DESIGN AND IMPLEMENTATION, 8., 2008, OSDI, San Diego, CA, USA. **Proceedings...** Berkeley, CA, USA: USENIX Association, 2008. P. 29–42.

ZHANG, S.; QIAN, Z.; WU, J.; LU, S. An Opportunistic Resource Sharing and Topology-Aware mapping framework for virtual networks. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATIONS, 31., 2012, INFOCOM, Orlando, FL, USA. **Proceedings...** Piscataway, NJ, USA: IEEE Press, 2012. P. 2408–2416. ISSN 0743-166X.

ZHANG, Y.; JUELS, A.; OPREA, A.; REITER, M. K. Homealone: Co-residency detection in the cloud via side-channel analysis. In: IEEE Symposium on Security and Privacy, 32., 2011, SP, Oakland, CA, USA. **Proceedings...** Washington, DC, USA: IEEE Computer Society, 2011. P. 313–328. ISBN 978-0-7695-4402-1.

ZIMMERMANN, P. R. **The official PGP user's guide**. Cambridge, MA, USA: MIT Press, 1995. ISBN 0-262-74017-6.

APPENDIX A RESUMO ESTENDIDO DA DISSERTAÇÃO

Computação em nuvem é um paradigma que possibilita aos locatários consumirem, sob demanda e de maneira elástica, recursos de hardware e software oferecidos por um provedor remoto. Nesse modelo, os provedores, com a finalidade de reduzir custos operacionais, oferecer escalabilidade de acordo com a demanda e atingir economias de escala, implementam *datacenters* de nuvem como ambientes compartilhados altamente multiplexados, com diferentes aplicações coexistindo sobre os mesmos recursos físicos (ARMBRUST et al., 2009, 2010).

Entretanto, não há uma abstração ou mecanismo disponível para capturar e controlar os requisitos de rede das interações entre as máquinas virtuais (VMs) alocadas (GUO et al., 2010a). Ademais, mecanismos como o controle de congestionamento do TCP e suas variantes (incluindo o *TCP-Friendly Rate Control* - TFRC - e o *Datagram Congestion Control Protocol* - DCCP -), apesar de serem escaláveis e proporcionarem alta utilização dos recursos, não isolam de forma robusta o tráfego de dados de aplicações distintas (ABTS; FELDERMAN, 2012b). Dessa forma, as VMs de uma mesma aplicação se comunicam em um ambiente (rede da nuvem) não controlado, compartilhado por todos os locatários. Consequentemente, locatários egoístas podem implementar suas aplicações de forma a levar vantagem indevida sobre outros usuários (ataques de interferência de desempenho), através da utilização de versões do TCP com controle de congestionamento mais brando, e locatários maliciosos podem lançar ataques de negação de serviço, por exemplo gerando tráfego espúrio dentro da nuvem (SHIEH et al., 2011).

Tais ataques prejudicam tanto locatários quanto provedores. Os locatários não possuem garantias de segurança e pagam pela utilização dos recursos contratados mesmo quando estão impossibilitados de utilizá-los devido a um ataque. Já os provedores possuem perda de receita, visto que a falta de disponibilidade da rede, causada por ataques de negação de serviço (*Denial of Service* - DoS), reduz o *throughput* da nuvem (SHIEH et al., 2011).

Com o objetivo de prover segurança e previsibilidade de desempenho no compartilhamento dos recursos da rede interna da nuvem, as abordagens presentes no estado-da-arte podem ser divididas em três classes principais: *i*) consolidação de VMs; *ii*) compartilhamento proporcional da rede; e *i*) virtualização de redes. Abordagens baseadas em consolidação de VMs (MENG; PAPPAS; ZHANG, 2010; WANG; MENG; ZHANG, 2011; BREITGAND; EPSTEIN, 2012) propõem adaptar

o problema de otimização do empacotamento estocástico (*Stochastic Bin Packing* - SBP) (KLEINBERG; RABANI; TARDOS, 1997; GOEL; INDYK, 1999) para lidar com a alocação de recursos em ambientes de nuvens computacionais. Contudo, tais abordagens não protegem a rede contra ataques de locatários egoístas e maliciosos.

Outra classe de trabalhos propõe o compartilhamento proporcional dos recursos de rede da nuvem de acordo com pesos atribuídos à entidades geradoras de tráfego (por exemplo, locatários, VMs e processos) (LAM et al., 2012; SHIEH et al., 2010, 2011). Apesar dos seus benefícios, tais propostas possuem alto custo de implementação (requerendo equipamentos específicos de rede) e alta sobrecarga de gerenciamento de recursos. Por fim, outra gama de trabalhos (GUO et al., 2010b, 2010a; BENSON et al., 2011; RODRIGUES et al., 2011; BALLANI et al., 2011b; XIE et al., 2012) propõe isolar diferentes aplicações (ou locatários) em redes virtuais distintas. Tal medida, entretanto, possui as desvantagens de acarretar em baixa utilização dos recursos de rede e em alta complexidade de gerenciamento de recursos. Tais desvantagens prejudicam a receita do provedor e, em última análise, são traduzidas em altos custos para os locatários.

Com base nos trabalhos presentes na literatura, considera-se, nesta dissertação, um modelo de ataque no qual um locatário egoísta e/ou malicioso possui os mesmos privilégios dos demais locatários da nuvem. Usuários egoístas lançam ataques de interferência de desempenho contra outras aplicações, com o objetivo de aumentar a vazão de rede das suas VMs. Já locatários maliciosos tem por objetivo realizar um ataque de DoS em um alvo previamente definido. Nesse caso, o locatário emprega os métodos descritos por Ristenpart et al. (2009) para posicionar as suas VMs próximas ao alvo. Consequentemente, as VMs desse locatário podem enviar uma quantidade suficiente de tráfego para sobrecarregar o destinatário (alvo), um enlace pertencente ao caminho ou um gargalo da rede (JENSEN et al., 2009).

Os ataques de interferência de desempenho e DoS são efetuados através de duas formas: *i*) pelo aumento do número de fluxos de dados, explorando a falta de isolamento entre fluxos devido ao uso do TCP (SHIEH et al., 2011); e *ii*) pelo envio de grandes fluxos de dados por meio do protocolo UDP. Considerando que a rede interna da nuvem é compartilhada por todos os locatários, o ataque pode afetar um grande número de aplicações.

Nesse contexto, é proposta uma estratégia de alocação de recursos para provedores de serviço de infraestrutura (*Infrastructure as a Service* - IaaS). Tal abordagem possibilita aumentar a segurança do compartilhamento dos recursos da rede interna da nuvem entre as aplicações de locatários por meio da redução do impacto de ataques de egoísmo e de locatários maliciosos. Diferentemente dos trabalhos presentes na literatura, investiga-se uma estratégia baseada no agrupamento de aplicações em domínios virtuais logicamente isolados, os quais são compostos por VMs e pela rede que as interconecta (Infraestruturas Virtuais - VIs).

Entretanto, existe um desafio fundamental a ser enfrentado: o problema de alocação de recursos em nuvem considerando enlaces de rede com largura de banda limitada é NP-Difícil (GUO et al., 2010b). Por essa razão, divide-se o problema em dois sub-problemas ou etapas menores, propõe-se uma estratégia de alocação para cada uma e combina-se os resultados. A primeira etapa é responsável por alocar as VIs no substrato físico (Função $\mathcal{G}$), enquanto a segunda distribui e mapeia as

aplicações no conjunto de infraestruturas virtuais (Função $\mathcal{F}$).

O agrupamento de aplicações em VIs possui dois benefícios, como segue. O primeiro é relacionado à segurança: o agrupamento provê isolamento entre aplicações provenientes de locatários mutuamente não-confiáveis. Ou seja, o sistema torna-se mais resiliente contra locatários que possam causar disrupção na rede, capturar informações confidenciais de outras aplicações ou utilizar uma quantidade maior de recursos para reduzir os seus custos em detrimento de outras aplicações. O segundo benefício diz respeito ao desempenho, já que o agrupamento permite à nuvem prover isolamento de desempenho entre aplicações com diferentes requisitos de largura de banda. Em suma, segurança e isolamento de desempenho aumentam as garantias de rede às aplicações, o que possibilita reduzir os custos dos locatários. Além disso, a estratégia proposta não requer a adição de novo hardware. Na verdade, ela pode ser implantada de duas maneiras: *i*) pela configuração dos dispositivos de rede; ou *ii*) pela modificação dos hypervisores de máquinas virtuais, de modo similar a Ballani et al. (2011b) e Xie et al. (2012).

Por outro lado, o número de grupos criados é pragmaticamente limitado pela sobrecarga introduzida pela tecnologia de virtualização. Ademais, o agrupamento tende a acarretar fragmentação de recursos ao alocar requisições de aplicações e afetar negativamente o grau de utilização dos recursos. Portanto, busca-se estudar diferentes estratégias de agrupamento, com base em relações de confiança mútua entre locatários e nos requisitos de rede das suas aplicações.

A.1 Contribuições

As principais contribuições apresentadas nesta dissertação se desdobram em três. Primeiro, é desenvolvida uma estratégia de alocação de recursos ciente de segurança e desempenho de rede para *datacenters* de nuvem que provêem serviços de IaaS. Tal esquema tem como objetivo principal aumentar a segurança e a previsibilidade de desempenho oferecidas às aplicações de locatários por meio do seu agrupamento em infraestruturas virtuais, com baixo impacto no grau de utilização dos recursos da nuvem.

Segundo, a estratégia é formalmente apresentada como um modelo de otimização de programação inteira mista (*Mixed-Integer Programming* - MIP), além de ser desenvolvida uma heurística construtiva para alocar eficientemente aplicações em nuvens de larga escala. A estratégia pode ser aplicada sobre várias topologias de rede de *datacenters*, tais como árvores com múltiplas raízes utilizadas atualmente (BALLANI et al., 2011b), VL2 (GREENBERG et al., 2009) e Fat-Tree (AL-FARES; LOUKISSAS; VAHDAT, 2008).

Por fim, a estratégia proposta é avaliada de duas maneiras. Primeiro, avalia-se o compromisso entre o ganho de segurança e desempenho para os locatários e o custo para os provedores (fragmentação de recursos) do esquema de alocação proposto. Os resultados mostram que a estratégia pode aumentar substancialmente a segurança e o desempenho com baixo custo extra. Em particular, a segurança aumenta com um comportamento logarítmico de acordo com o número de VIs, enquanto a fragmentação de recursos cresce linearmente de acordo com o aumento

do número de VIs oferecidas pelo provedor. Segundo, a receita do provedor é avaliada com base em dois modelos de cobrança: *i*) o modelo atual utilizado por provedores de nuvem pública (por exemplo, Amazon), no qual somente recursos computacionais são cobrados; e *ii*) um modelo que considera também a cobrança pela utilização dos recursos de rede, similar ao utilizado por trabalhos relacionados (BALLANI et al., 2011b; XIE et al., 2012). No primeiro, a receita é reduzida em torno de 5% com a utilização da estratégia proposta nessa dissertação em comparação com o modelo atual, enquanto no segundo a receita é aumentanda em até 20%.

A.2 Trabalhos Futuros

Como trabalhos futuros, considera-se a melhoria da estratégia de alocação de recursos em quatro direções principais. Primeiramente, planeja-se incluir requisitos de segurança e de mapeamento de enlaces virtuais em múltiplos caminhos físicos na etapa de alocação das VIs no substrato físico (Função $\mathcal{G}$). Dessa forma, pode-se aumentar o isolamento entre as infraestruturas virtuais no substrato físico e a resiliência em caso de falhas. Segundo, visa-se desenvolver novos algoritmos de alocação para tratar elasticidade horizontal (ou seja, a habilidade de adicionar ou remover dinamicamente VMs em uma aplicação), o que irá possibilitar o crescimento elástico das aplicações durante os seus ciclos de execução.

Em terceiro lugar, busca-se aumentar a segurança e minimizar a fragmentação de recursos por meio da expansão ou redução dinâmica do tamanho das VIs, considerando-se as características das requisições de aplicações que chegam. Quarto, planeja-se desenvolver metaheurísticas para otimizar o uso dos recursos da infraestrutura da nuvem de acordo com o passar do tempo. Para tal, torna-se necessário considerar a migração das VMs previamente alocadas, considerando-se as vantagens e desvantagens tanto para provedores quanto para locatários.

APPENDIX B FUNCTION $\mathcal{G}$ PROGRAMMING MODEL

```
/* Model parameters */

# Number of virtual infrastructures
param num_ivs, integer, >= 1;

/* Sets */

# Enumeration of virtual infrastructures
set Iv, default {1..num_ivs};

# VI graphs
set Mv {i in Iv};                           # VMs from VI i
set Rv {i in Iv};                           # ToRs (racks) from Vi i
set Swv {i in Iv};                          # Remaining switches from Vi i
set Sv {i in Iv} := Rv[i] union Swv[i];     # All switches from Vi i
set Nv {i in Iv} := Mv[i] union Sv[i];      # All nodes from Vi i
set Ev {i in Iv}, within Nv[i] cross Nv[i]; # Links from Vi i

# Physical substrate graph
set Ms;                     # Servers
set Ss;                     # Switches
set Ns := Ms union Ss;      # All nodes
set Es, within Ns cross Ns; # Links

/* Input */

# Virtual infrastructures
param Bandv {i in Iv, (u,v) in Ev[i]};
param Oversubv {i in Iv, (u,v) in Ev[i]};
#param Bandri {i in Iv};

# Physical substrate
param Bands {(u,v) in Es};
param Slots {ms in Ms};
param Costs {ss in Ss};
param Caps {ss in Ss};

# Parameters
param alpha {(u,v) in Es}, > 0, <= 1;
param gamma, >= 0;
param delta, >= 0;

/* Variables */

var x {i in Iv, sv in Sv[i], ss in Ss}, binary;
var y {i in Iv, (w1,w2) in Ev[i], (w3,w4) in Es}, binary;
var z {i in Iv, mv in Mv[i], ms in Ms}, binary;

/* Optimization objective */

minimize Z:
sum{(w1,w2) in Es} (
    # Cost bw
```

```
    gamma * ((alpha[w1,w2] / Bands[w1,w2]) * sum{i in Iv, (w3,w4) in Ev[i]} y[i,w3,
        w4,w1,w2] * Bandv[i,w3,w4])
) + delta * sum {ss in Ss}(
    # Cost sw
    sum{i in Iv, sv in Sv[i]} x[i,sv,ss] * Costs[ss]
);

/* Constraints */

s.t. vms_constr {i in Iv, mv in Mv[i]}:
        sum{ms in Ms} z[i,mv,ms] = 1;

s.t. sw_constr {i in Iv, sv in Sv[i]}:
        sum{ss in Ss} x[i,sv,ss] = 1;

s.t. vm_cap_constr {ms in Ms}:
        sum{i in Iv, mv in Mv[i]} z[i,mv,ms] <= Slots[ms];

s.t. sw_cap_constr {ss in Ss}:
        sum{i in Iv, sv in Sv[i]} x[i,sv,ss] <= Caps[ss];

s.t. sw_cap_constr2 {i in Iv, ss in Ss}:
        sum{sv in Sv[i]} x[i,sv,ss] <= 1;

s.t. link_cap_constr {(w1,w2) in Es}:
        sum{i in Iv, (w3,w4) in Ev[i]} y[i,w3,w4,w1,w2] * Bandv[i,w3,w4] <= Bands[
            w1,w2];

s.t. substrate_path_constr {i in Iv, w3 in Ss, w1 in Sv[i], w2 in Sv[i]: (w1,w2) in
     Ev[i]}:
        sum{w4 in Ss: w3 <> w4 and (w3,w4) in Es} y[i,w1,w2,w3,w4] - sum{w4 in Ss:
            w3 <> w4 and (w4,w3) in Es} y[i,w1,w2,w4,w3] = x[i,w1,w3] - x[i,w2,w3];

s.t. link_tor_constr {i in Iv, ss in Ss, mv in Mv[i], sv in Rv[i]: (sv,mv) in Ev[i
    ]}:
        x[i,sv,ss] = sum{ms in Ms: (ss,ms) in Es} y[i,mv,sv,ms,ss];

s.t. vm_link_constr {i in Iv, ms in Ms, ss in Ss, mv in Mv[i], sv in Rv[i]: (sv,mv)
     in Ev[i] and (ss,ms) in Es}:
        z[i,mv,ms] = y[i,mv,sv,ms,ss];

s.t. symm_link_constr {i in Iv, (w1,w2) in Ev[i], (w3,w4) in Es}:
        y[i,w1,w2,w3,w4] - y[i,w2,w1,w4,w3] = 0;

solve;

end;
```

APPENDIX C FUNCTION $\mathcal{F}$ PROGRAMMING MODEL

```
/* Model parameters */

# Number of application requests
param num_apps, integer, >= 1;

# Number of virtual infrastructures
param num_ivs, integer, >= 1;

/* Sets */

# Enumeration of applications
set Ar, default {1..num_apps};

# Enumeration of virtual infrastructures
set Iv, default {1..num_ivs};

# VI graphs
set Mv {i in Iv};                                  # VMs from VI i
set Rv {i in Iv};                                  # ToRs (racks) from Vi i
set Swv {i in Iv};                                 # Remaining switches
set Sv {i in Iv} := Rv[i] union Swv[i];            # All switches
set Nv {i in Iv} := Mv[i] union Sv[i];             # All nodes
set Ev {i in Iv}, within Nv[i] cross Nv[i];        # Links

# Sets of rack pairs
set P {i in Iv} := {r1 in Rv[i], r2 in Rv[i]: r1 <> r2};


/* Tenant applications */

# Application requests
param Mra {a in Ar};
param Bandra {a in Ar};
param Trust {a1 in Ar, a2 in Ar};

# Virtual infrastructures
param Bandv {i in Iv, (u,v) in Ev[i]};
param Oversubv {i in Iv, (u,v) in Ev[i]};

# Balancing parameters
param gamma, >= 0;
param delta, >=0;

/* Variables */

var f {a in Ar, i in Iv, (w1, w2) in Ev[i]}, >= 0;
var g {a in Ar, i in Iv, r in Rv[i], m in Mv[i]: (r,m) in Ev[i]}, binary;
var h {a in Ar, i in Iv, (w1, w2) in Ev[i], (r1,r2) in P[i]}, binary;
var F {a in Ar, i in Iv, (w1, w2) in Ev[i], (r1,r2) in P[i]}, >= 0;
var B {a in Ar, i in Iv, (r1,r2) in P[i]}, binary;
var G {a in Ar, i in Iv}, binary;
var H {a1 in Ar, a2 in Ar, i in Iv}, binary;
```

```
/* Optimization objective */

minimize Z: gamma * sum{a1 in Ar, a2 in Ar, i in Iv} ((1-Trust[a1,a2]) * H[a1,a2,i]
     * Mra[a1] * Bandra[a1]) + delta * (sum{a in Ar, i in Iv, (w1,w2) in Ev[i]} f[a
    ,i,w1,w2]);

/* Constraints */

s.t. trust_iv_constr1 {a1 in Ar, a2 in Ar, i in Iv: a1 <> a2}:
            H[a1,a2,i] <= G[a1,i];

s.t. trust_iv_constr2 {a1 in Ar, a2 in Ar, i in Iv: a1 <> a2}:
            H[a1,a2,i] <= G[a2,i];

s.t. trust_iv_constr3 {a1 in Ar, a2 in Ar, i in Iv: a1 <> a2}:
            H[a1,a2,i] >= G[a1,i] + G[a2,i] - 1;

s.t. bw_iv_constr3 {a in Ar, i in Iv}:
         G[a,i] = (sum{r in Rv[i], m in Mv[i]: (r,m) in Ev[i]} g[a,i,r,m]) / Mra[a
             ];

s.t. vms_ar_constr {a in Ar}:
        sum{i in Iv, r in Rv[i], m in Mv[i]: (r,m) in Ev[i]} g[a,i,r,m] = Mra[a];

s.t. vm_iv_constr {i in Iv, r in Rv[i], m in Mv[i]: (r,m) in Ev[i]}:
        sum{a in Ar} g[a,i,r,m] <= 1;

s.t. vm_iv_constr3 {a in Ar, i in Iv, r in Rv[i], m in Mv[i]: (r,m) in Ev[i]}:
        g[a,i,r,m] <= G[a,i];

s.t. cap_link_constr {i in Iv, (w1,w2) in Ev[i]}:
        sum{a in Ar} f[a,i,w1,w2] <= Bandv[i,w1,w2] * Oversubv[i,w1,w2];

s.t. flow_rack_constr1 {a in Ar, i in Iv, (r1,r2) in P[i]}:
        sum{w2 in Sv[i]: (r1,w2) in Ev[i]} F[a,i,r1,w2,r1,r2] >=
                ((sum{m in Mv[i]: (r1,m) in Ev[i]} g[a,i,r1,m]) * Bandra[a]) - ((
                    Mra[a]/2) * Bandra[a] * B[a,i,r1,r2]);

s.t. flow_rack_constr2 {a in Ar, i in Iv, (r1,r2) in P[i]}:
        sum{w2 in Sv[i]: (r1,w2) in Ev[i]} F[a,i,r1,w2,r1,r2] >=
                ((sum{m in Mv[i]: (r2,m) in Ev[i]} g[a,i,r2,m]) * Bandra[a]) - ((
                    Mra[a]/2) * Bandra[a] * (1 - B[a,i,r1,r2]));

s.t. flow_rack_constr3 {a in Ar, i in Iv, (r1,r2) in P[i]}:
        (sum{m in Mv[i]: (r1,m) in Ev[i]} g[a,i,r1,m]) * Bandra[a] <=
                (sum{m in Mv[i]: (r2,m) in Ev[i]} g[a,i,r2,m]) * Bandra[a] + (Mra[a
                    ]/2) * Bandra[a] * B[a,i,r1,r2];


s.t. flow_rack_constr4 {a in Ar, i in Iv, (r1,r2) in P[i]}:
        (sum{m in Mv[i]: (r2,m) in Ev[i]} g[a,i,r2,m]) * Bandra[a] <=
                 (sum{m in Mv[i]: (r1,m) in Ev[i]} g[a,i,r1,m]) * Bandra[a] + (Mra[
                     a]/2) * Bandra[a] * (1 - B[a,i,r1,r2]);

s.t. flow_rack_constr5 {a in Ar, i in Iv, (r1,r2) in P[i]}:
        sum{w2 in Sv[i]: (r1,w2) in Ev[i]} F[a,i,r1,w2,r1,r2] <=
                (sum{m in Mv[i]: (r1,m) in Ev[i]} g[a,i,r1,m]) * Bandra[a] + (Mra[a
                    ]/2) * Bandra[a] * B[a,i,r1,r2];

s.t. flow_rack_constr6 {a in Ar, i in Iv, (r1,r2) in P[i]}:
        sum{w2 in Sv[i]: (r1,w2) in Ev[i]} F[a,i,r1,w2,r1,r2] <=
                (sum{m in Mv[i]: (r2,m) in Ev[i]} g[a,i,r2,m]) * Bandra[a] + (Mra[a
                    ]/2) * Bandra[a] * (1 - B[a,i,r1,r2]);

s.t. flow_rack_constr7 {a in Ar, i in Iv, (r1,r2) in P[i]}:
        sum{w1 in Sv[i]: (w1,r2) in Ev[i]} F[a,i,w1,r2,r1,r2] =
            sum{w2 in Sv[i]: (r1,w2) in Ev[i]} F[a,i,r1,w2,r1,r2];

s.t. flow_network_constr {a in Ar, i in Iv, (r1,r2) in P[i], w3 in Sv[i]:  w3 <> r1
     and w3 <> r2}:
        sum{w4 in Sv[i]: w4 <> w3 and w4 <> r1 and (w3,w4) in Ev[i]} F[a,i,w3,w4,r1
            ,r2] -
        sum{w4 in Sv[i]: w4 <> w3 and w4 <> r2 and (w4,w3) in Ev[i]} F[a,i,w4,w3,r1
```

```
            ,r2] = 0;

s.t. flow_path_constr {a in Ar, i in Iv, (r1,r2) in P[i], w1 in Sv[i], w2 in Sv[i]:
     (w1,w2) in Ev[i]}:
        F[a,i,w1,w2,r1,r2] <= Bandv[i,w1,w2] * h[a,i,w1,w2,r1,r2];

s.t. unsplittable_flow_constr {a in Ar, i in Iv, w1 in Sv[i], (r1,r2) in P[i]}:
        sum{w2 in Sv[i]: (w1,w2) in Ev[i]} h[a,i,w1,w2,r1,r2] <= 1;

s.t. unsplittable_flow_constr2 {a in Ar, i in Iv, w1 in Rv[i], w2 in Sv[i], (r1,r2)
     in P[i]: w1 <> w2 and (w1,w2) in Ev[i]}:
         h[a,i,w1,w2,r1,r2] + h[a,i,w1,w2,r2,r1] <= 1;

s.t. app_flow_constr {a in Ar, i in Iv, w1 in Sv[i], w2 in Sv[i]: (w1,w2) in Ev[i
    ]}:
        f[a,i,w1,w2] = sum{(r1,r2) in P[i]} F[a,i,w1,w2,r1,r2];

s.t. flow_tor_constr {a in Ar, i in Iv, w1 in Mv[i], w2 in Rv[i]: (w1,w2) in Ev[i
    ]}:
        f[a,i,w1,w2] = g[a,i,w2,w1] * Bandra[a];

s.t. symm_flow_constr {a in Ar, i in Iv, w1 in Mv[i], w2 in Rv[i]: (w1,w2) in Ev[i
    ]}:
        f[a,i,w1,w2] = f[a,i,w2,w1];

solve;

end;
```

APPENDIX D PUBLISHED PAPER AT SBSEG

In this appendix, we present the paper "Mitigando Ataques de Egoísmo e Negação de Serviço em Nuvens via Agrupamento de Aplicações" published at SBSeg 2012. This paper presents an initial approach to address the lack of security in the intra-cloud network.

- Title: Mitigando Ataques de Egoísmo e Negação de Serviço em Nuvens via Agrupamento de Aplicações
- Conference: XII Brazilian Symposium on Information and Computer System Security (SBSeg 2012)
- URL: http://sbseg2012.ppgia.pucpr.br/en/index.php
- Date: November 19 – 22, 2012
- Location: Curitiba, PR, Brazil

Mitigando Ataques de Egoísmo e Negação de Serviço em Nuvens via Agrupamento de Aplicações

Daniel Stefani Marcon, Miguel Cardoso Neves, Rodrigo Ruas Oliveira, Luciana Salete Buriol, Luciano Paschoal Gaspary, Marinho Pilla Barcellos

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

{daniel.stefani, mcneves, ruas.oliveira, buriol, paschoal, marinho}@inf.ufrgs.br

Resumo. *Na computação em nuvem, locatários consomem, sob demanda, recursos de hardware e software oferecidos por um provedor remoto. Entretanto, o compartilhamento da rede interna da nuvem por todos os locatários, aliado à falta de isolamento entre fluxos de dados decorrente do uso dos protocolos TCP e UDP, possibilita a ocorrência de ataques de egoísmo e negação de serviço. Os algoritmos de alocação atuais não impedem que a disponibilidade dos recursos de rede seja afetada por ataques. Este artigo propõe uma estratégia para a alocação de aplicações de locatários que visa mitigar o impacto de ataques de egoísmo e negação de serviço na rede interna da nuvem. A ideia chave, inédita na literatura científica, consiste no agrupamento de aplicações em infraestruturas virtuais considerando níveis de confiança mútua entre os locatários. Resultados de avaliações demonstram que a estratégia proposta é capaz de oferecer proteção contra ataques de egoísmo e negação de serviço com pouco ou nenhum custo extra.*

Abstract. *Cloud computing is a model where tenants consume on-demand hardware and software resources from a remote provider. However, the sharing of the internal network by all tenants, combined to the lack of data flow isolation due to the use of TCP and UDP, allows the occurrence of selfish and denial of service attacks. The current allocation algorithms do not prevent the availability of network resources to be affected by such attacks. In this paper, we propose a strategy for the allocation of tenants applications, which aims at mitigating the impact of selfish and denial of service attacks in the cloud internal network. The key, novel idea is to group applications into virtual infrastructures considering the mutual trust between pairs of tenants. Evaluation results show that the proposed strategy is able to offer protection against attacks of selfishness and DoS with little or no extra cost.*

1. Introdução

Computação em nuvem é um paradigma que possibilita aos locatários consumirem, sob demanda e de maneira elástica, recursos de hardware e software oferecidos por um provedor remoto. Nesse modelo, os provedores, com a finalidade de reduzir custos operacionais e oferecer escalabilidade de acordo com a demanda, implementam *datacenters* de nuvem como ambientes compartilhados altamente multiplexados, com diferentes aplicações coexistindo sobre os mesmos recursos físicos [Armbrust et al. 2010].

Entretanto, não há uma abstração ou mecanismo disponível para capturar os requisitos de rede das interações entre as máquinas virtuais (VMs) alocadas [Guo et al. 2010]. Ademais, mecanismos como o controle de congestionamento do TCP e suas variantes (incluindo TFRC e DCCP), apesar de serem escaláveis e proporcionarem alta utilização dos recursos, não isolam de forma robusta o tráfego de dados de aplicações distintas [Abts and Felderman 2012]. Dessa forma, as VMs de uma mesma aplicação se comunicam em um ambiente (rede da nuvem) não controlado, compartilhado por todos os locatários. Consequentemente, locatários egoístas podem implementar suas aplicações de forma a levar vantagem indevida sobre outros usuários, através da utilização de versões do TCP com controle de congestionamento mais brando, e locatários maliciosos podem lançar ataques de negação de serviço, por exemplo gerando tráfego espúrio dentro da nuvem [Shieh et al. 2011].

Tais ataques prejudicam tanto locatários quanto provedores. Os locatários pagam pela utilização dos recursos contratados mesmo quando estão impossibilitados de utilizá-los devido a um ataque. Já os provedores possuem perda de receita [Greenberg et al. 2009], visto que a falta de disponibilidade da rede, causada por ataque de DoS, reduz o *throughput* da nuvem [Shieh et al. 2011].

Visando alocar recursos com garantias de disponibilidade de rede, as abordagens presentes na literatura focam em consolidação de VMs [Breitgand and Epstein 2012, Meng et al. 2010, Wang et al. 2011] ou no compartilhamento da rede baseado em pesos [Shieh et al. 2011, Lam and Varghese 2010]. Entretanto, elas não impedem ataques de egoísmo e de DoS, visto que a utilização dos recursos da rede de uma aplicação depende das demais aplicações da nuvem.

Nesse contexto, o presente artigo propõe uma estratégia de alocação de recursos para provedores de IaaS (*Infrastructure as a Service*), apresentada como um modelo de otimização, que visa mitigar ataques de egoísmo e de comportamento malicioso na rede interna da nuvem. Diferentemente de trabalhos anteriores, investiga-se uma estratégia baseada no agrupamento de aplicações em infraestruturas virtuais[1] (VIs), que são capazes de oferecer um certo nível de isolamento entre aplicações. São estudadas diferentes estratégias de agrupamento de locatários, considerando o nível de confiança mútuo e os requisitos de rede (largura de banda) das suas aplicações.

O agrupamento visa prover isolamento do tráfego de rede entre aplicações provenientes de locatários mutuamente não confiáveis. Além disso, o agrupamento permite fornecer isolamento entre o tráfego de aplicações com requisitos de largura de banda distintos, minimizando a interferência nociva que pode ser causada ao misturar tráfego com diferentes características [Shieh et al. 2011, Abts and Felderman 2012]. Além da proteção contra os ataques supracitados, o isolamento/agrupamento propiciado pela estratégia proposta pode ser benéfico contra outros tipos de ataques, como ataques contra a privacidade dos dados dos locatários [Ristenpart et al. 2009]. Os mesmos serão explorados em trabalhos futuros. O presente artigo tem as seguintes contribuições:

- discussão e avaliação do impacto de ataques de egoísmo e de negação de serviço no contexto de redes internas de nuvens;
- proposta de uma estratégia inédita para alocação de recursos, baseada no agrupamento de aplicações em VIs considerando níveis de confiança mútua

[1] O termo infraestrutura virtual é utilizado para representar uma rede virtual com as suas máquinas virtuais.

entre os locatários, e aplicável a diferentes topologias de datacenters, tais como Fat-Tree [Greenberg et al. 2009] e VL2 [Al-Fares et al. 2008];
- avaliação analítica da estratégia proposta perante diferentes níveis de agrupamento e a comparação da mesma com um esquema padrão sem agrupamento.

O restante desse artigo está organizado da seguinte maneira. A Seção 2 apresenta os principais trabalhos relacionados. A Seção 3 define o modelo de ataque considerado e a Seção 4 define as formulações básicas utilizadas no restante do artigo. Na Seção 5, a estratégia de alocação de recursos é apresentada. Na Seção 6, o modelo de alocação proposto é avaliado e, por fim, as considerações finais são apresentadas na Seção 7.

2. Trabalhos Relacionados

As redes internas das nuvens atuais são compartilhadas por todos os locatários, honestos ou não. Apesar disso, não há um mecanismo disponível para capturar e controlar os recursos de rede utilizados pelas VMs alocadas [Grobauer et al. 2011].

Nesse sentido, Liu [Liu 2010] descreve como um atacante pode obter informações sobre a topologia da rede da nuvem, com a finalidade de realizar um ataque de negação de serviço. Já Ristenpart et al. [Ristenpart et al. 2009] conduziram um estudo de caso na Amazon EC2 (*Elastic Compute Cloud*), mostrando diversas vulnerabilidades do ambiente. Os autores relatam que não possuem nenhum dado privilegiado da nuvem, tendo conhecimento apenas das informações divulgadas publicamente pela Amazon. Desse modo, eles assumem que os atacantes são clientes comuns da nuvem. Eles demonstram que a EC2 pode ser mapeada de acordo com as suas zonas de disponibilidade e que é possível verificar a co-residência entre VMs, inclusive listando diversas possibilidades de efetuar tal verificação. O referido trabalho mostra que um locatário egoísta e/ou malicioso, conforme abordado no presente artigo, pode realizar diversos tipos de ataques à nuvem, inclusive de DoS na rede.

Apesar dessas vulnerabilidades, os algoritmos de alocação de recursos empregados atualmente em nuvens utilizam *round-robin* entre servidores ou entre *racks*, considerando somente recursos computacionais (processamento, memória e armazenamento) [Kitsos et al. 2012]. Contudo, torna-se necessário considerar também os recursos de rede, a fim de mitigar ataques de egoísmo e de comportamento malicioso.

O problema de segurança na rede interna da nuvem não é facilmente resolvível. Nesse contexto, a alocação de recursos em nuvens ciente dos recursos de rede representa um grande desafio de pesquisa. As abordagens presentes na literatura focam em consolidação de VMs ou no compartilhamento da rede por meio de pesos. Abordagens baseadas em consolidação de VMs [Meng et al. 2010, Wang et al. 2011, Breitgand and Epstein 2012] propõem que a alocação seja executada por meio da adaptação do problema de otimização *bin packing* [Goel and Indyk 1999]. Contudo, elas não impedem que o tráfego de aplicações egoístas ou maliciosas interfira no tráfego de aplicações honestas. Já Shieh et al. [Shieh et al. 2011] e Lam e Varghese [Lam and Varghese 2010] propõem um esquema de compartilhamento de largura de banda baseado em pesos. Esse esquema, porém, não impede que aplicações maliciosas realizem ataques de egoísmo e de negação de serviço na rede, visto que a largura de banda utilizada por uma aplicação na rede é dependente das outras aplicações da nuvem.

Em outra gama de trabalhos, é explorada a alocação de tarefas em grades computacionais respeitando restrições de confiança entre a tarefa e os recursos da grade computacional [Costa and Carmo 2007]. A estratégia apresentada nesse artigo, em uma perspectiva distinta, explora as relações de confiança mútuas entre os locatários da nuvem.

De forma geral, os recursos da rede interna de *datacenters* frequentemente representam o gargalo quando comparados aos recursos computacionais [Greenberg et al. 2009]. Esse gargalo, aliado à falta de mecanismos para o controle do compartilhamento de recursos na rede, tende a facilitar ataques de egoísmo e comportamento malicioso. Nesse sentido, a estratégia apresentada nesse artigo visa mitigar tais ataques, sendo ciente tanto de recursos de rede quanto de recursos computacionais. Adicionalmente, a utilização de VIs diminui a eficácia dos métodos utilizados por Ristenpart et al. [Ristenpart et al. 2009] para a verificação de co-residência, visto que aplicações mutuamente não confiáveis são alocadas em redes virtuais distintas.

3. Modelo de Ataque

Similarmente à Ristenpart et al. [Ristenpart et al. 2009], considera-se que um locatário egoísta e/ou malicioso possui os mesmos privilégios dos demais locatários da nuvem. Usuários egoístas utilizam versões mais agressivas do TCP (*i.e.*, versões fora dos padrões, cujo código foi modificado para realizar controle de congestionamento mais brando) com o objetivo de aumentar a vazão de rede das suas VMs. Já locatários maliciosos tem por objetivo realizar um ataque de DoS em um alvo previamente definido. Nesse caso, o locatário emprega os métodos descritos por Ristenpart et al. [Ristenpart et al. 2009] para posicionar as suas VMs próximas ao alvo. Consequentemente, as VMs desse locatário podem enviar uma quantidade suficiente de tráfego para sobrecarregar o destinatário (alvo), um enlace pertencente ao caminho ou um gargalo da rede [Jensen et al. 2009]. No presente trabalho, considera-se somente o ataque de um locatário. Entretanto, ataques de conluio são possíveis, através de múltiplas identidades.

Os ataques de egoísmo e de DoS são efetuados através do aumento do número de fluxos de dados, explorando a falta de isolamento entre fluxos devido ao uso do TCP [Shieh et al. 2011], e pelo envio de grandes fluxos por meio do protocolo UDP. Considerando que a rede interna da nuvem é compartilhada por todos os locatários, o ataque pode afetar um grande número de aplicações.

O ataque precisa ser lançado por um "insider", ou seja, um locatário registrado na nuvem, e que pode portanto ser em tese responsabilizado. Entretanto, não há requisitos estritos para criar contas em nuvens e, mesmo que houvesse, o ataque poderia ser efetuado através de uma conta comprometida [Greenberg et al. 2008]. Além disso, a detecção de ataques não é simples. Tráfego malicioso pode simular um tráfego honesto, dificultando a distinção entre os dois tipos, e os esquemas de ofuscação, utilizados pelos provedores para ocultar a localização dos recursos na nuvem, não são capazes de parar um adversário persistente [Shieh et al. 2011].

4. Modelo de Sistema

Com base nas questões de segurança apresentadas anteriormente, é proposta uma estratégia na forma de um modelo de otimização para a alocação de aplicações de

locatários que visa mitigar o impacto de ataques de egoísmo e negação de serviço na rede interna da nuvem. A execução desses ataques é facilitada atualmente na nuvem, visto que os provedores não cobram pelo tráfego de dados na rede interna [Guo et al. 2010].

Esta seção define as formulações básicas utilizadas para a modelagem da estratégia. As notações são representadas por meio das seguintes regras: *i*) sobrescritos s, v e r indicam entradas relacionadas à rede do substrato físico da nuvem, às infraestruturas virtuais e às requisições dos locatários, respectivamente; *ii*) subscritos são índices provenientes de atributos, variáveis ou elementos de um conjunto. O sistema de notação utilizado é similar ao empregado por Chowdhury et al. [Chowdhury et al. 2009].

4.1. Requisições de Aplicações

Cada requisição de aplicação $a \in A^r$, proveniente de um locatário, é definida pela tupla $(M_a^r,\ Band_a^r)$. O número de máquinas virtuais requisitadas é representado por M_a^r. Para facilidade de exposição do modelo desenvolvido, os detalhes das VMs são abstraídos e todas elas são consideradas iguais (consomem a mesma quantidade de recursos de CPU, memória e armazenamento). De modo similar a Ballani et al. [Ballani et al. 2011], uma requisição é estendida para especificar, além de recursos computacionais, recursos de rede. A largura de banda disponível para uma VM comunicar-se com outra VM da mesma aplicação é indicada por $Band_a^r$, que é um número real positivo ($Band_a^r \in \mathbb{R}^+$).

Ademais, cada aplicação $a \in A^r$ possui ou não confiança nas outras aplicações da nuvem, de acordo com as relações mútuas entre os locatários das mesmas. A confiança é representada por $T^r_{a_i,a_j}$, que indica se a aplicação a_i confia na aplicação a_j. Com o objetivo de simplificar a avaliação (mas sem prejuízo ao modelo), no presente trabalho assume-se que as relações de confiança são diretas, binárias e simétricas. Em outras palavras, um locatário confia ou não em um outro com quem ele interage e, se confia, então é recíproco; presume-se que um esquema padrão como PGP [Zimmermann 1995] pode ser usado para o estabelecimento de confiança mútua.

4.2. Infraestruturas Virtuais

O conjunto de infraestruturas virtuais é indicado por I^v. Cada elemento $i \in I^v$ é representado por um grafo bidirecional valorado $G_i^v = (S_i^v, M_i^v, E_i^v, Band^v, Oversub^v)$, sendo seus elementos definidos a seguir. O conjunto de dispositivos de rede (*switches*) de i é indicado por S_i^v. O conjunto de máquinas virtuais de i, tipicamente entre 20 e 40 por *rack* [Greenberg et al. 2009], é indicado por M_i^v e o conjunto de enlaces virtuais por E_i^v. Cada enlace $e^v = (w_i, w_j) \in E_i^v$ conecta os nodos w_i e w_j ($w_i, w_j \in S_i^v \cup M_i^v$). Além disso, cada enlace $e^v \in E_i^v$ possui uma largura de banda bidirecional $Band^v(e^v) \in \mathbb{R}^+$ e um fator de sobrecarga $Oversub^v(e^v) \in \mathbb{Z}^+$ empregado pelo provedor aos recursos de rede. Além disso, o subconjunto de *switches Top-of-Rack* (ToR), retornado pela função $ToR(S_i^v)$, é representado por R_i^v ($R_i^v \subset S_i^v$).

4.3. Infraestrutura Física

De forma similar a Guo et al. [Guo et al. 2010], o substrato físico da nuvem é modelado considerando servidores, dispositivos de rede e enlaces. Ele é representado por

um grafo bidirecional valorado $G^s = (S^s\ , M^s\ , E^s\ , Band^s\ , Slots^s\ , Cost^s\ , Cap^s)$, no qual S^s é conjunto de dispositivos de rede (*switches*), M^s é o conjunto de servidores e E^s é o conjunto de enlaces. Cada servidor $m^s \in M^s$ possui um número de *slots*[2] ($Slots^s(m^s) \in \mathbb{Z}^+$). Cada *switch* $s^s \in S^s$ possui um número máximo de *switches* virtuais que ele pode hospedar ($Cap^s(s^s) \in \mathbb{Z}^+$) e possui um custo ($Cost^s(s^s) \in \mathbb{R}^+$) para hospedar um *switch* virtual. O custo é proporcional à importância do *switch* na rede, ou seja, *switches* localizados em níveis mais próximos do núcleo da rede possuem maior custo para hospedar *switches* virtuais. Cada enlace $e^s = (w_i, w_j) \in E^s \mid w_i, w_j \in S^s \cup M^s$ entre os nodos w_i e w_j é associado a uma capacidade (largura de banda) bidirecional $Band^s(e^s) \in \mathbb{R}^+$.

5. Estratégia de Alocação de Recursos

A estratégia apresentada nesse artigo para a alocação de recursos visa mitigar ataques de egoísmo e comportamento malicioso na rede interna da nuvem. Diferentemente de trabalhos anteriores, o objetivo é alcançado por meio do agrupamento de aplicações em infraestruturas virtuais considerando a confiança mútua entre os locatários e a minimização do tráfego de rede gerado pelas interações entre as VMs da mesma aplicação.

Entretanto, existe um desafio fundamental a ser enfrentado: o problema de alocação de recursos em nuvem considerando enlaces de rede com largura de banda limitada é NP-Difícil [Guo et al. 2010]. Por essa razão, dividimos o problema em dois sub-problemas ou etapas menores, propomos uma estratégia de alocação ótima para cada um, e combinamos os resultados. A primeira etapa distribui e mapeia as aplicações em infraestruturas virtuais, enquanto a segunda é responsável por alocar cada infraestrutura virtual no substrato físico da nuvem.

Neste trabalho, o algoritmo para a formação do conjunto de infraestruturas virtuais (I^v) é abstraído e foca-se na alocação de aplicações nas VIs segundo critérios de segurança e desempenho e no mapeamento das VIs no substrato físico. Dessa forma, o conjunto de VIs é utilizado como entrada para o modelo de otimização. Apesar disso, sugere-se um um critério para formação de VIs, que poderia ser usado pelo provedor. As VIs podem ser formadas com base em informações obtidas a partir de serviços de monitoramento de recursos da nuvem (por exemplo, o Amazon CloudWatch[3]). O uso de informações pertencentes ao histórico da utilização de recursos sobre um grande período de tempo tende a facilitar e a melhorar a qualidade do processo de alocação [Meng et al. 2010]. Além disso, a topologia de rede de cada VI não é restrita pela topologia da infraestrutura física, visto que a heterogeneidade de topologias pode ser tratada no mapeamento das VIs no substrato físico [Chowdhury et al. 2009].

De modo geral, o processo completo é representado na Figura 1, que indica cada etapa por meio de uma função. A primeira etapa é representada por $\mathcal{F} : A^r \rightarrow I^v$, enquanto a segunda é denotada por $\mathcal{G} : I^v \rightarrow G^s$. A função $\mathcal{H} : A^r \rightarrow G^s$ realiza o mapeamento direto de aplicações na infraestrutura física da nuvem, mostrando como esse processo é efetuado atualmente em nuvens. Dessa forma, pode-se dizer que $\mathcal{H}$ é uma composição de $\mathcal{F}$ e $\mathcal{G}$.

[2]Como assume-se que todas as VMs são iguais, ou seja, consomem a mesma quantidade de recursos dos servidores, os *slots* também são considerados iguais.

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

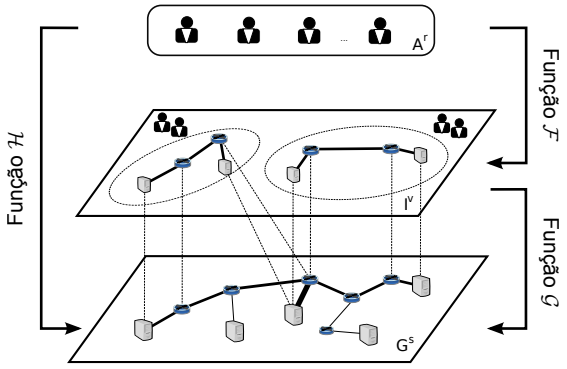


Figura 1. O processo de alocação de recursos na nuvem.

Apesar da estratégia ser composta por duas etapas, ambas podem ser executadas simultaneamente, visto que a saída de uma não interfere na entrada da outra. Isso implica que o tempo necessário para a conclusão do processo é o tempo de execução da etapa mais longa. Nesta seção, primeiramente é apresentada a função $\mathcal{F}$ e, após, a função $\mathcal{G}$.

5.1. Mapeamento de Aplicações em Infraestruturas Virtuais

A função $\mathcal{F}$ tem por objetivo mapear as aplicações em VIs considerando os níveis mútuos de confiança entre os locatários das mesmas. A função de mapeamento utiliza como dados de entrada as requisições de aplicações na nuvem (Seção 4.1) e o conjunto de infraestruturas virtuais (Seção 4.2).

As principais variáveis relacionadas ao modelo de otimização são:

- $F_{a,i,(w_1,w_2),p} \in \mathbb{R}^+$: indica o tamanho do fluxo total de dados para a aplicação $a \in A^r$ no enlace $(w_1, w_2) \in E_i^v \mid w_1, w_2 \in S_i^v$ para a comunicação entre o par de *racks* $p = (r_1, r_2) \mid r_1, r_2 \in R_i^v$ e $r_1 \neq r_2$ da VI $i \in I^v$. O valor desse fluxo, definido com base no número de VMs da aplicação a nos *racks* r_1 e r_2, será explicado posteriormente (Equação 2);
- $g_{a,i,r,m} \in \mathbb{B}$: indica se a VM $m \in M_i^v$ do *rack* $r \in R_i^v$ da VI $i \in I^v$ foi alocada para a aplicação $a \in A^r$;
- $G_{a,i}$: indica se a aplicação $a \in A^r$ está na VI $i \in I^v$;
- $h_{a,i,(w_1,w_2),p} \in \mathbb{B}$: indica se a aplicação $a \in A^r$ utiliza o enlace $(w_1, w_2) \in E_i^v \mid w_1, w_2 \in S_i^v$ para a comunicação entre o par de *racks* $p = (r_1, r_2) \mid r_1, r_2 \in R_i^v$ e $r_1 \neq r_2$ da VI $i \in I^v$;
- H_{i,a_1,a_2}: indica se as aplicações $a_1, a_2 \in A^r$ estão alocadas na VI $i \in I^v$.

A função objetivo do problema de otimização dessa etapa, apresentada na Equação 1, busca minimizar uma penalização imposta por alocar aplicações provenientes de locatários mutuamente não confiáveis na mesma VI. A penalização considera as características das aplicações (número de VMs e largura de banda) mutuamente não confiáveis. Por exemplo, a aplicação a_1 possui 20 VMs, cada uma com largura de banda de 200 Mbps. Caso a aplicação a_2 não confie em a_1, a função objetivo gera uma penalização de a_1 em a_2, cujo valor é $20 \times 200 = 4000$. Essa fórmula também endereça as diferenças entre as aplicações dos locatários. Ou seja, há maior risco de um locatário atacar outro, se não houver confiança entre ambos, caso ele requisite uma aplicação composta por 50 VMs e 100 Mbps de largura de banda para cada VM em detrimento de uma aplicação composta por uma única VM com 100 Mbps.

$$Z = \text{Min} \sum_{i \in I^v} \sum_{a_1 \in A^r} \sum_{a_2 \in A^r} ((1 - T^r_{a_1,a_2}) * H_{i,a_1,a_2} * M^r_{a_1} * Band^r_{a_1}) \quad (1)$$

As restrições do problema especificam o mapeamento das aplicações nas VIs, de acordo com os recursos computacionais e de rede disponíveis. Na rede, o tráfego de comunicação entre VMs localizadas no mesmo *rack* não possui custo, visto que ele permanece interno ao *rack* e só utiliza os enlaces que ligam as VMs ao *switch* ToR. Entretanto, o tráfego entre VMs de *racks* distintos possui custo, que é definido pela largura de banda consumida e pelos enlaces utilizados.

Desse modo, as VMs de cada aplicação são alocadas em *grupos*, de modo semelhante a Ballani et al. [Ballani et al. 2011]. Um grupo é composto pelo conjunto de VMs da mesma aplicação localizadas no mesmo *rack*. Por isso, visa-se criar o menor número possível de grupos de VMs, ou seja, alocar as VMs da mesma aplicação próximas umas das outras, o que reduz o tráfego de dados na rede e tende a dificultar ataques de egoísmo e DoS.

Com essa abordagem, é necessário garantir que haja largura de banda disponível para a comunicação entre os grupos de VMs. Por exemplo, considerando a existência de dois grupos $G_{a,1}$ e $G_{a,2}$ de VMs da aplicação $a \in A^r$ e que a comunicação ocorra em um caminho virtual P^v, todos os enlaces desse caminho devem possuir a largura de banda necessária disponível. Uma única VM da aplicação a não pode enviar ou receber dados a uma taxa maior que a requisitada ($Band^r_a$). Dessa forma, o tráfego entre dois grupos é limitado à largura de banda exigida pelo menor dos grupos: $\min(|G_{a,1}|, |G_{a,2}|) * Band^r_a$. A Figura 2 exemplifica uma situação em que duas aplicações possuem dois grupos de VMs cada uma. Nesse caso, é necessário garantir a largura de banda na rede para a comunicação entre a dupla de grupos de VMs de cada aplicação. A restrição para a comunicação de um grupo com os outros grupos da aplicação a é formulada da seguinte maneira:

$$B_{a,g_i} = \min \left(|g_i| * Band^r_a, \sum_{g \in G^r_a, g \neq g_i} |g| * Band^r_a \right) \quad \forall g_i \in G^r_a \quad (2)$$

onde G^r_a é o conjunto de grupos da aplicação a.

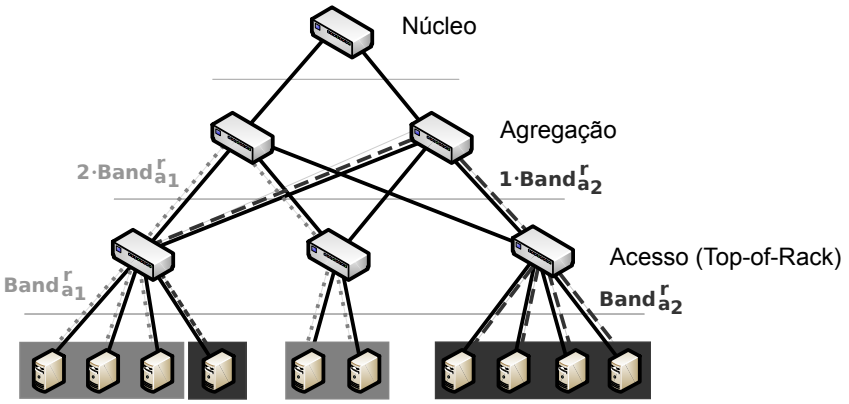


Figura 2. Comunicação em rede entre grupos de VMs.

Em *datacenters*, o protocolo *Spanning Tree* assegura que o tráfego entre máquinas dentro da mesma camada dois da rede é encaminhado ao longo de uma árvore de expansão. Os dispositivos de rede da nuvem geralmente são interconectados com uma malha de enlaces, cuja utilização é balanceada através do uso de múltiplos caminhos de mesmo custo (*Equal-Cost Multipath* - ECMP) [Abts and Felderman 2012]. Dada a quantidade de multiplexação sobre a malha de enlaces e o número limitado de caminhos, os múltiplos caminhos podem ser considerados como um único caminho agregado para o tratamento de questões de largura de banda.Dessa forma, o modelo matemático proposto considera a utilização de um caminho para a comunicação entre os grupos de VMs da mesma aplicação. As fórmulas para o cálculo dos caminhos são mostradas nas Equações 3 e 4, que determinam o tráfego de saída do *rack* fonte e de entrada no *rack* destino, e nas Equações 5 e 6, utilizadas para garantir que um fluxo possui somente um caminho na rede. Outras restrições do modelo, que buscam garantir que a capacidade dos recursos virtuais não seja excedida, são omitidas devido a questões de espaço.

$$\sum_{\substack{w_2 \in S_i^v | \\ (r_1,w_2) \in E_i^v}} F_{a,i,(r_1,w_2),p} = \sum_{\substack{w_1 \in S_i^v | \\ (w_1,r_2) \in E_i^v}} F_{a,i,(w_1,r_2),p} = \min \left(\sum_{m \in M_i^v} g_{a,i,r_1,m}, \sum_{m \in M_i^v} g_{a,i,r_2,m} \right) * Band_a^r$$
$$\forall i \in I^v, \quad \forall a \in A^r, \quad \forall r_1, r_2 \in R_i^v \mid r_1 \neq r_2 \text{ and } p = (r_1, r_2) \quad (3)$$

$$\sum_{w_4 \in S_i^v \setminus w_3 | (w_3,w_4) \in E_i^v} F_{a,i,(w_3,w_4),p} - \sum_{w_4 \in S_i^v \setminus w_3 | (w_4,w_3) \in E_i^v} F_{a,i,(w_4,w_3),p} = 0$$
$$\forall i \in I^v, \quad \forall a \in A^r, \quad \forall p = (r_1, r_2) \mid r_1, r_2 \in R_i^v \text{ and } r_1 \neq r_2, \quad \forall w_3 \in S_i^v \setminus \{r_1, r_2\} \quad (4)$$

$$F_{a,i,(w_1,w_2),p} \leq Band^v(w_1, w_2) * h_{a,i,(w_1,w_2),p}$$
$$\forall a \in A^r, \quad \forall i \in I^v, \quad \forall w_1, w_2 \in S_i^v \mid (w_1, w_2) \in E_i^v, \quad \forall p = (r_1, r_2) \mid r_1, r_2 \in R_i^v \text{ and } r_1 \neq r_2 \quad (5)$$

$$\sum_{w_2 \in S_i^v | (w_1,w_2) \in E_i^v} h_{a,i,(w_1,w_2),p} \leq 1$$
$$\forall a \in A^r, \quad \forall i \in I^v, \quad \forall w_1 \in S_i^v, \quad \forall p = (r_1, r_2) \mid r_1, r_2 \in R_i^v \text{ and } r_1 \neq r_2 \quad (6)$$

5.2. Mapeamento de Infraestruturas Virtuais no Substrato Físico

A função $\mathcal{G}$ é responsável pelo mapeamento das VIs no substrato físico da nuvem. Ela aborda um problema similar ao problema de mapeamento de redes virtuais (*virtual network embedding* - VNE), o que possibilita lidar com a heterogeneidade das topologias das VIs em relação ao substrato físico [Chowdhury et al. 2009]. No entanto, além da topologia da rede (composta por dispositivos de rede e enlaces), ela considera também a utilização de nodos computacionais (servidores e máquinas virtuais).

O problema de otimização desenvolvido para essa etapa utiliza como entradas o conjunto de infraestruturas virtuais (Seção 4.2) e a infraestrutura física da nuvem (Seção 4.3). Além disso, são utilizados os parâmetros $\alpha_{(w_1,w_2)}$ e γ, como segue. O fator $\alpha_{(w_1,w_2)}$ é definido de acordo com a importância do enlace (w_1, w_2) e possui

valor máximo 1 ($0 < \alpha_{(w_1,w_2)} \leq 1$). Já o parâmetro γ é utilizado para balancear os dois componentes da função objetivo.

As seguintes variáveis são utilizadas no modelo:

- $x_{i,s^v,s^s} \in \mathbb{B}$: indica se o *switch* virtual $s^v \in S_i^v$ da infraestrutura virtual $i \in I^v$ está alocado no *switch* físico $s^s \in S^s$;
- $y_{i,e^v,(w_1,w_2)} \in \mathbb{B}$: indica se o enlace virtual $e^v \in E_i^v$, pertencente à infraestrutura virtual $i \in I^v$, utiliza o enlace físico $(w_1, w_2) \in E^s$;
- $z_{i,m^v,m^s} \in \mathbb{B}$: indica se a VM $m^v \in M_i^v$ da infraestrutura virtual $i \in I^v$ está alocada no servidor $m^s \in M^s$.

A função objetivo (7) busca minimizar os recursos utilizados para alocar o conjunto de infraestruturas virtuais. O primeiro componente está relacionado aos enlaces da rede. Ao dividir-se $\alpha_{(w_1,w_2)}$ pela largura de banda do enlace, garante-se que os enlaces com menor prioridade e maior capacidade são preferidos em detrimento dos enlaces com maior importância. O segundo componente quantifica o custo de alocar *switches* virtuais em *switches* do substrato físico. Apesar da função não considerar requisitos de segurança, planeja-se incluir critérios de resiliência/segurança em trabalhos futuros, a fim de permitir maior isolamento entre as VIs no substrato físico e alocar enlaces virtuais em múltiplos caminhos, mitigando possíveis ataques de DoS.

$$Z = \text{Min} \sum_{(w_1,w_2) \in E^s} \frac{\alpha_{(w_1,w_2)}}{Band^s(w_1,w_2)} * \sum_{i \in I^v} \sum_{e^v \in E_i^v} y_{i,e^v,(w_1,w_2)} * Band^v(e^v) + \gamma * \sum_{s^s \in S^s} \sum_{i \in I^v} \sum_{s^v \in S_i^v} x_{i,s^v,s^s} * Cost^s(s^s) \tag{7}$$

Os valores das variáveis pertencentes à função objetivo são definidos com base nas restrições do modelo. Inicialmente, é necessário assegurar a capacidade dos servidores (Equação 8), dos *switches* físicos (Equação 9) e dos enlaces do substrato (Equação 10) em suportar os respectivos elementos virtuais.

$$\sum_{i \in I^v} \sum_{m^v \in M_i^v} z_{i,m^v,m^s} \leq Slots^s(m^s) \qquad \forall m^s \in M^s \tag{8}$$

$$\sum_{i \in I^v} \sum_{s^v \in S_i^v} x_{i,s^v,s^s} \leq Cap^s(s^s) \qquad \forall s^s \in S^s \tag{9}$$

$$\sum_{i \in I^v} \sum_{e^v \in E_i^v} (y_{i,e^v,(w_1,w_2)} * Band^v(e^v)) \leq Band^s(e^s) \qquad \forall e^s = (w_1,w_2) \in E^s \tag{10}$$

Também deve-se garantir que os enlaces virtuais são mapeados em caminhos válidos do substrato físico (Equação 11). As demais restrições do modelo, responsáveis pelo mapeamento dos recursos virtuais no substrato físico, são omitidas devido a questões de espaço.

$$\sum_{w_4 \in S^s} y_{i,e^v,(w_3,w_4)} - \sum_{w_4 \in S^s} y_{i,e^v,(w_4,w_3)} = x_{i,w_1,w_3} - x_{i,w_2,w_3}$$
$$\forall i \in I^v, \quad \forall w_1, w_2 \in S_i^v \mid e^v = (w_1, w_2) \in E_i^v, \quad \forall w_3 \in S^s \quad (11)$$

6. Avaliação

Nesta seção, são avaliados dois aspectos da estratégia de alocação de recursos proposta. Primeiramente, a qualidade da solução é quantificada de duas formas: o benefício propiciado, com o aumento do nível de segurança às aplicações, e o custo disso, decorrente das restrições impostas na alocação pelas relações de confiança entre locatários. Em segundo lugar, busca-se verificar o tempo de processamento necessário à alocação ótima de recursos, o que afeta a viabilidade da solução.

6.1. Ambiente de Avaliação

Os experimentos foram implementados e executados no IBM ILOG CPLEX *Optimization Studio*[4]. Todos os experimentos foram executados em um computador Intel Core i3-2120 de 3.30 GHz, com 8 GB de memória RAM e sistema operacional GNU/Linux Debian x86_64. Cada experimento é limitado a um tempo máximo de duração de 1 hora e com uma árvore de busca de no máximo 4 GB, devido à complexidade do problema. Dessa forma, os resultados mostrados são os melhores resultados factíveis alcançados pelo CPLEX no período estipulado.

De modo similar a trabalhos relacionados [Ballani et al. 2011, Guo et al. 2010, Shieh et al. 2011], o substrato físico da nuvem foi definido como uma topologia em árvore. O substrato físico é composto por 80 servidores, cada um com 4 *slots*, divididos igualmente em 10 *racks*. Os *racks* são conectados entre si por *switches* das camadas superiores com diversidade de caminho.

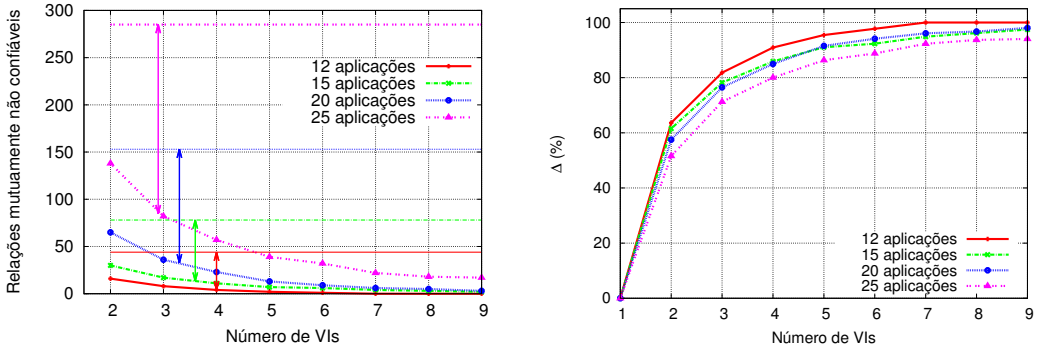
A carga de trabalho a ser alocada são conjuntos de 12, 15, 20 ou 25 requisições para instanciar aplicações na nuvem, cada uma pertencendo a um locatário distinto. O número de VMs e a largura de banda especificados em cada requisição são uniformemente distribuídos, respectivamente, nos intervalos [2, 10] e [100,450] Mbps. A confiança mútua entre locatários foi gerada através das relações diretas entre os mesmos em um grafo aleatório, com grau de cada vértice (locatário) seguindo uma distribuição $P(k) \propto \frac{1}{k}$.

6.2. Qualidade da Solução

A qualidade da solução é quantificada de acordo com o critério de confiança mútua entre os locatários das aplicações da nuvem. Ou seja, busca-se avaliar a solução com base na segurança oferecida pelo agrupamento de aplicações de locatários em comparação com o cenário atual, no qual não é efetuado nenhum tipo de agrupamento. Refletindo isso, a métrica empregada é o número de relações entre aplicações de locatários que não possuem uma relação de confiança entre si e que foram alocadas no mesmo agrupamento. Deseja-se que esse valor seja o menor possível, pois representa uma certa exposição da aplicação a ataques.

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

A Figura 3(a) mostra a variação do número de relações mutuamente não confiáveis de acordo com a quantidade de VIs oferecidas pelo provedor para os diferentes conjuntos de requisições de aplicações. Com base nesses dados, define-se diferença média do nível de confiança no cenário com agrupamento: $\Delta = 1 - \frac{\Phi}{\Gamma}$, onde Φ representa o número de relações não confiáveis entre aplicações dentro do agrupamento e Γ denota o número de relações não confiáveis sem o agrupamento. O valor de Δ é apresentado na Figura 3(b), que mostra que o número de aplicações não é o fator principal para o aumento da segurança, mas sim o número de VIs oferecidas pelo provedor. De modo geral, é possível verificar que o ganho de segurança tem um comportamento logarítmico de acordo com o número de VIs.



(a) Relações não confiáveis.

(b) Diferença do nível de segurança com o agrupamento.

Figura 3. Segurança com o agrupamento de aplicações na nuvem.

O agrupamento pode ser benéfico também em relação ao consumo de largura de banda e ao isolamento entre o tráfego de aplicações com requisitos de banda distintos, como segue. As VMs pertencentes a uma aplicação são posicionadas, por definição, dentro de uma VI, enquanto os recursos de uma VI tendem a ser alocados próximos uns dos outros na infraestrutura física. Portanto, quanto menor a VI, maior é a chance de VMs comunicantes estarem próximas e, consequentemente, consumirem menos recursos de rede. O isolamento entre o tráfego de aplicações com requisitos de banda distintos, por sua vez, minimiza a interferência nociva que pode ser causada ao misturar tráfego com diferentes características [Abts and Felderman 2012].

Por outro lado, a criação de VIs e o agrupamento de locatários nas mesmas pode ter dois efeitos negativos. Primeiro, a criação e a manutenção das VIs introduz uma sobrecarga de gerência e comunicação em relação a uma rede convencional. Em relação a isso, segundo trabalhos anteriores, essa sobrecarga é proporcionalmente pequena e plenamente justificável considerando os benefícios [Ballani et al. 2011].

Segundo, o agrupamento tende a restringir a alocação de aplicações na nuvem, por causa da fragmentação dos recursos. Ou seja, quanto menor as VIs criadas, maior a fragmentação (interna). Com isso, maior será o número de requisições que não poderão ser atendidas apesar da capacidade agregada disponível. No presente trabalho, não quantificamos esse efeito porque modelamos o problema de otimização considerando recursos suficientes para atender todas as requisições, similarmente à Guo et al. 2010. Pretendemos analisar mais a fundo essa questão, em cenários com carência de recursos, em trabalhos futuros.

6.3. Tempo de Processamento para Alocação

Outra métrica analisada na avaliação da estratégia de alocação de recursos é o tempo de processamento necessário à alocação de um conjunto de aplicações a um conjunto de VIs, considerando o tratamento *offline* neste artigo. A complexidade da função $\mathcal{F}$ é combinatória de acordo com o número de aplicações, o número de VIs e os seus respectivos elementos virtuais. A complexidade da função $\mathcal{G}$ é combinatória de acordo com as VIs oferecidas e o substrato físico. Por isso, o espaço de busca por soluções de ambos os problemas de otimização é grande, sendo necessária uma grande quantidade de tempo de processamento e memória para encontrar a solução ótima. Apesar disso, soluções factíveis, mas não ótimas, podem ser encontradas com menor dificuldade.

Nesse contexto, a complexidade dos dois problemas serve como grande motivação para o desenvolvimento, em trabalhos futuros, de meta-heurísticas (p.ex., *Simmulated Annealing*) e heurísticas construtivas considerando a chegada dinâmica (*online*) de requisições de aplicações.

7. Considerações Finais

As economias de escala estão impulsionando aplicações distribuídas a coexistir em infraestruturas compartilhadas. Entretanto, a falta de mecanismos para controlar o compartilhamento de recursos da rede interna da nuvem possibilita a ocorrência de ataques de egoísmo e DoS, prejudicando tanto provedores quanto locatários. Desse modo, a alocação de aplicações considerando recursos de rede e critérios de segurança representa um grande desafio de pesquisa.

Este artigo apresentou uma estratégia de alocação de recursos que visa mitigar ataques de egoísmo e negação de serviço na rede interna da nuvem. Este é o primeiro trabalho na literatura a explorar o agrupamento de aplicações em infraestruturas virtuais considerando a confiança mútua entre os locatários. Dessa forma, o sistema torna-se mais resiliente contra aplicações que usariam uma fatia desproporcional de recursos, com ou sem o intuito de prejudicar outras aplicações. A estratégia foi avaliada analiticamente perante diferentes níveis de agrupamento e comparada com um esquema padrão, sem agrupamento.

Como trabalhos futuros, pretende-se desenvolver meta-heurísticas e heurísticas construtivas, que considerem alta taxa de chegada e saída de locatários (*churn*) e elasticidade horizontal, para o modelo proposto. Também pretende-se acrescentar questões referentes à segurança na função $\mathcal{G}$ e explorar o isolamento/agrupamento propiciado pela estratégia proposta para mitigar outros tipos de ataques (p.ex., privacidade).

Referências

Abts, D. and Felderman, B. (2012). A guided tour of data-center networking. *Commun. ACM*, 55(6):44–51.

Al-Fares, M., Loukissas, A., and Vahdat, A. (2008). A scalable, commodity data center network architecture. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, SIGCOMM '08, pages 63–74, New York, NY, USA. ACM.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53:50–58.

Ballani, H., Costa, P., Karagiannis, T., and Rowstron, A. (2011). Towards predictable datacenter networks. In *Proceedings of the ACM SIGCOMM 2011 conference on SIGCOMM*, SIGCOMM '11, pages 242–253, New York, NY, USA. ACM.

Breitgand, D. and Epstein, A. (2012). Improving Consolidation of Virtual Machines with Risk-aware Bandwidth Oversubscription in Compute Clouds. In *Proceedings of the 31th conference on Information communications*, INFOCOM'12, Piscataway, NJ, USA. IEEE Press.

Chowdhury, N., Rahman, M., and Boutaba, R. (2009). Virtual network embedding with coordinated node and link mapping. In *INFOCOM 2009, IEEE*, pages 783 –791.

Costa, R. B. and Carmo, L. F. R. C. (2007). Avaliação de Confiança Contextual em Grades Computacionais Multímodo usando Plataformas Seguras. In *VII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais, SBSeg 2007*, pages 173–185.

Goel, A. and Indyk, P. (1999). Stochastic load balancing and related problems. In *Foundations of Computer Science, 1999. 40th Annual Symposium on*, pages 579 –586.

Greenberg, A., Hamilton, J., Maltz, D. A., and Patel, P. (2008). The cost of a cloud: research problems in data center networks. *SIGCOMM Comput. Commun. Rev.*, 39(1):68–73.

Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P., and Sengupta, S. (2009). Vl2: a scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, pages 51–62, New York, NY, USA. ACM.

Grobauer, B., Walloschek, T., and Stocker, E. (2011). Understanding Cloud Computing Vulnerabilities. *Security Privacy, IEEE*, 9(2):50–57.

Guo, C., Lu, G., Wang, H. J., Yang, S., Kong, C., Sun, P., Wu, W., and Zhang, Y. (2010). Secondnet: a data center network virtualization architecture with bandwidth guarantees. In *Proceedings of the 6th International Conference*, Co-NEXT '10, pages 15:1–15:12, New York, NY, USA. ACM.

Jensen, M., Schwenk, J., Gruschka, N., and Iacono, L. L. (2009). On Technical Security Issues in Cloud Computing. In *Cloud Computing, 2009. CLOUD '09. IEEE International Conference on*, pages 109–116.

Kitsos, I., Papaioannou, A., Tsikoudis, N., and Magoutis, K. (2012). Adapting data-intensive workloads to generic allocation policies in cloud infrastructures. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 25 –33.

Lam, T. and Varghese, G. (2010). NetShare: Virtualizing Bandwidth within the Cloud. Technical report cs2010-0957, University of California.

Liu, H. (2010). A new form of DOS attack in a cloud and its avoidance mechanism. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, CCSW '10, pages 65–76, New York, NY, USA. ACM.

Meng, X., Pappas, V., and Zhang, L. (2010). Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pages 1154–1162, Piscataway, NJ, USA. IEEE Press.

Ristenpart, T., Tromer, E., Shacham, H., and Savage, S. (2009). Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 199–212, New York, NY, USA. ACM.

Shieh, A., Kandula, S., Greenberg, A., Kim, C., and Saha, B. (2011). Sharing the data center network. In *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, NSDI'11, pages 23–23, Berkeley, CA, USA. USENIX Association.

Wang, M., Meng, X., and Zhang, L. (2011). Consolidating virtual machines with dynamic bandwidth demand in data centers. In *INFOCOM, 2011 Proceedings IEEE*, pages 71 –75.

Zimmermann, P. R. (1995). *The official PGP user's guide*. MIT Press, Cambridge, MA, USA.

APPENDIX E PUBLISHED PAPER AT IFIP NETWORKING

In this appendix, we present the paper "Trust-based Grouping for Cloud Datacenters: improving security in shared infrastructures" submitted to IFIP Networking 2013 conference.

- Title: Trust-based Grouping for Cloud Datacenters: improving security in shared infrastructures
- Conference: 12th IFIP Networking 2013
- URL: http://networking2013.poly.edu/
- Date: May 22 – 24, 2013
- Location: Brooklyn, New York, USA

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].

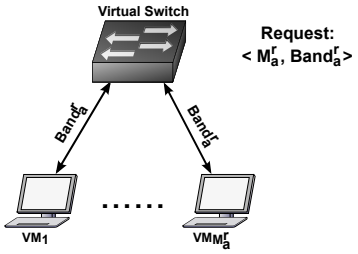


Fig. 1: Tenant's view of an application's network topology.

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.

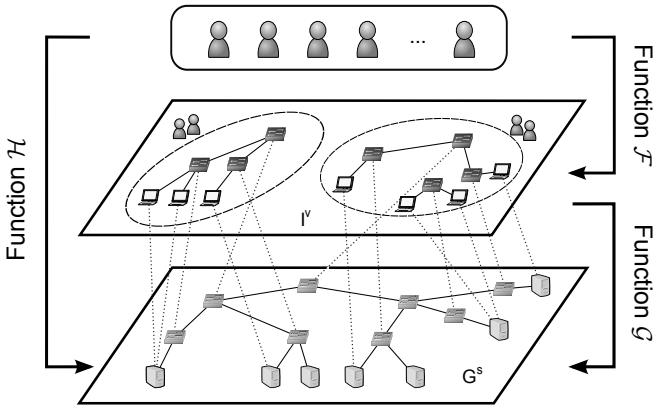


Fig. 2: Cloud resource allocation overview.

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 $e^v = (w_1, w_2) \in E_i^v,\ \forall i \in I^v$:

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

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:

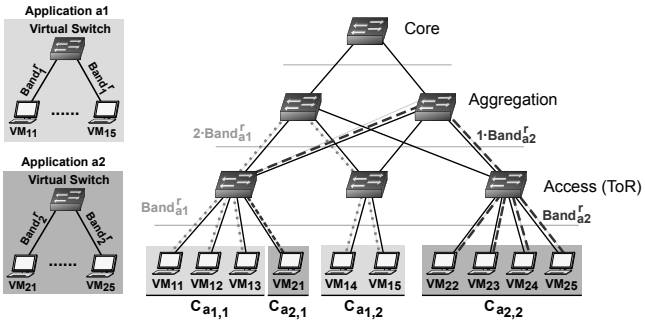


Fig. 3: Communication between VM clusters.

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.

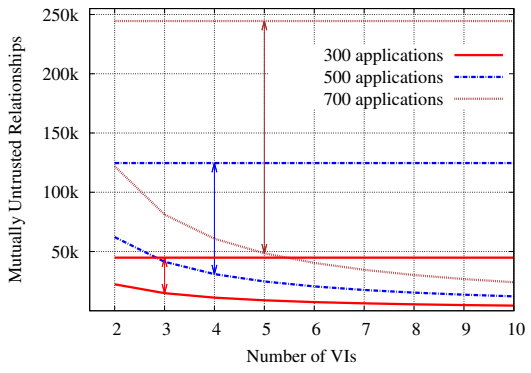


Fig. 4: Security when allocating batches of applications.

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.

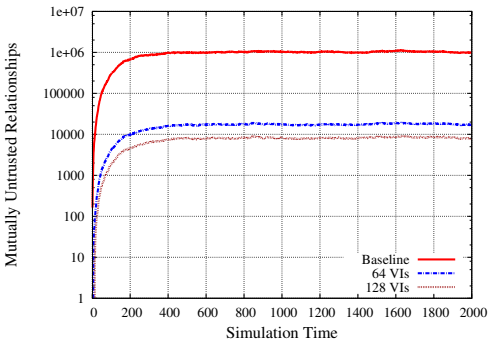


Fig. 5: Security in an online setting.

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.

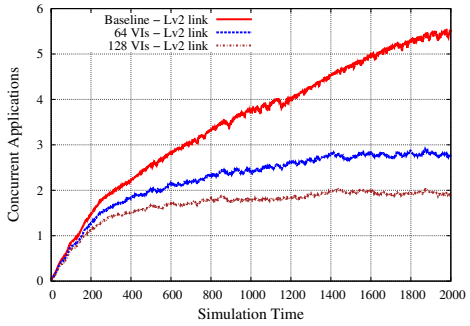


Fig. 6: Mean number of applications sharing a level-2 link.

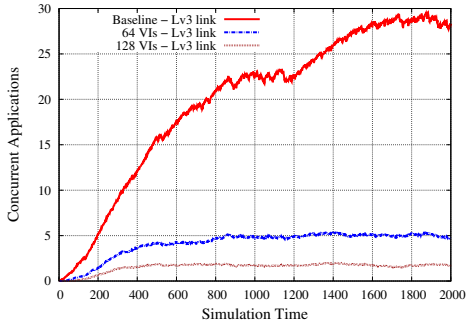


Fig. 7: Mean number of applications sharing a level-3 link.

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.

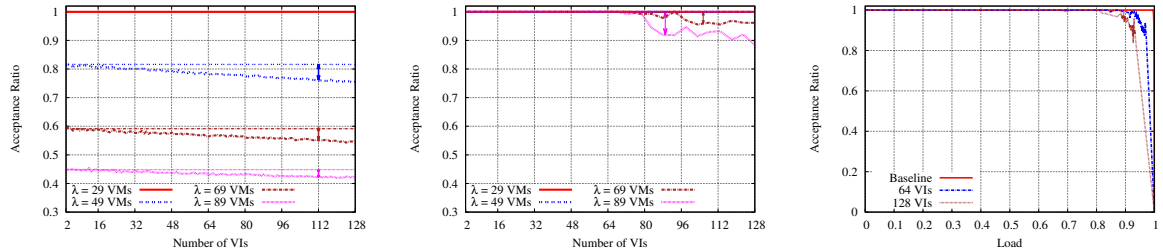
Figure 8(b) depicts the acceptance ratio with cloud-load between 70% and 80% (i.e., the usual load of public cloud platforms, such as Amazon EC2 [27]). We see that the acceptance ratio decreases according to the number of VIs offered and the size of applications (that is, the bigger the applications 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.

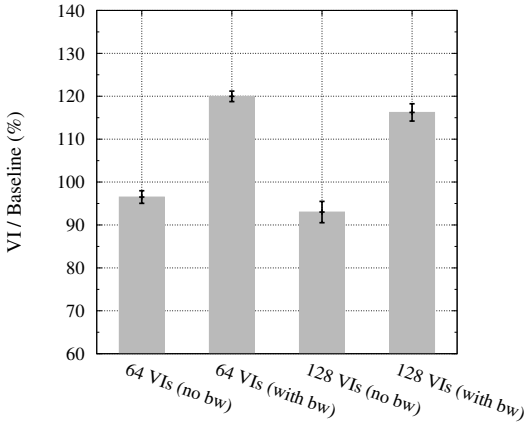


Fig. 9: Provider revenue.

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.

APPENDIX F OTHER PAPERS

Authored papers:

- **MARCON, D. S.**; BAYS, L. R. Balanceamento de Carga Orientado a Fluxos de Dados: Otimizando a Utilização de Servidores Web. In: ESCOLA REGIONAL DE REDES DE COMPUTADORES, 9., 2011, ERRC, São Leopoldo, RS, Brazil. Proceedings... Porto Alegre, RS, BR: SBC, 2011. p. 1 – 4.

- **MARCON, D. S.**; BAYS, L. R. Flow Based Load Balancing: Optimizing Web Servers Resource Utilization. In: Journal of Applied Computing Research (JACR), v. 1, p. 76 – 83, 2012.

Coauthored papers:

- OLIVEIRA, R. R.; BAYS, L. R.; **MARCON, D. S.**; NEVES, M. C.; BURIOL, L. S.; GASPARY, L. P.; BARCELLOS, M. P. Redes Virtuais Seguras: Uma Nova Abordagem de Mapeamento para Proteger contra Ataques de Disrupção na Rede Física. In: SIMPÓSIO BRASILEIRO DE SEGURANÇA DA INFORMAÇÃO E DE SISTEMAS COMPUTACIONAIS, 12., 2012, SBSEG, Curitiba, PR, Brazil. Proceedings... Porto Alegre, RS, BR: SBC, 2012. P. 235 – 248.

- OLIVEIRA, R. R.; BAYS, L. R.; **MARCON, D. S.**; NEVES, M. C.; BURIOL, L. S.; GASPARY, L. P.; BARCELLOS, M. P. DoS-Resilient Virtual Networks through Multipath Embedding and Opportunistic Recovery. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 28., 2013, SAC, Coimbra, Portugal. Proceedings... New York, NY, USA: ACM, 2009. P.1 – 6.

- OLIVEIRA, R. R.; **MARCON, D. S.**; BAYS, L. R.; NEVES, M. C.; BURIOL, L. S.; GASPARY, L. P.; BARCELLOS, M. P. No More Backups: Toward Efficient Embedding of Survivable Virtual Networks. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS, 12., 2013, ICC, Budapest, Hungary. Proceedings... Piscataway, NJ, USA: IEEE Press, 1983. P. 1 – 5.

- NEVES, M. C.; **MARCON, D. S.**; OLIVEIRA, R. R.; BAYS, L. R.; BURIOL, L. S.; GASPARY, L. P.; BARCELLOS, M. P. IoNCloud: uma abordagem não entrópica orientada a tráfego para reserva e isolamento de re-

cursos em nuvens. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS, 31., 2013, SBRC, Brasília, DF, Brazil. Proceedings.... Porto Alegre, RS, BR: SBC, 2013. P. 1 – 14.