QVIA-SDN: Towards QoS-Aware Virtual Infrastructure Allocation on SDN-based Clouds

Felipe Rodrigo de Souza, Charles Christian Miers, Adriano Fiorese, Marcos Dias de Assunção, Guilherme Piêgas Koslovski

To cite this version:

Felipe Rodrigo de Souza, Charles Christian Miers, Adriano Fiorese, Marcos Dias de Assunção, Guilherme Piêgas Koslovski. QVIA-SDN: Towards QoS-Aware Virtual Infrastructure Allocation on SDN-based Clouds. Journal of Grid Computing, Springer Verlag, 2019, 10.1007/s10723-019-09479-x . hal-02107009

HAL Id: hal-02107009
https://hal.archives-ouvertes.fr/hal-02107009

Submitted on 23 Apr 2019

Noname manuscript No.
(will be inserted by the editor)

QVIA-SDN: Towards QoS-Aware Virtual Infrastructure Allocation on SDN-based Clouds

Felipe Rodrigo de Souza · Charles Christian Miers · Adriano Fiorese · Marcos Dias de Assunção · Guilherme Piegas Koslovski

Received: date / Accepted: date

Abstract Virtual Infrastructures (VIs) emerged as a potential solution for network evolution and cloud services provisioning on the Internet. Deploying VIs, however, is still challenging mainly due to a rigid management of networking resources. By splitting control and data planes, Software-Defined Networks (SDN) enable custom and more flexible management, allowing for reducing data center usage, as well as providing mechanisms to guarantee bandwidth and latency control on switches and endpoints. However, reaping the benefits of SDN for VI embedding in cloud data centers is not trivial. Allocation frameworks require combined information from the control plan (*e.g.,* isolation policies, flow identification) and data (*e.g.,* storage capacity, flow table configuration) to find a suitable solution. In this context, the present work proposes a mixed integer programming formulation for the VI allocation problem that considers the main challenges regarding SDN-based cloud data centers. Some constraints are then relaxed resulting in a linear program, for which a heuristic is introduced. Experimental results of the mechanism, termed as QVIA-SDN, highlight that an SDN-aware allocation solution can reduce the data center usage and improve the quality-of-service perceived by hosted tenants.

Felipe Rodrigo de Souza
Graduate Program in Applied Computing – Santa Catarina State University – Joinville, Brazil
E-mail: dcc6frs@joinville.udesc.br

Charles Christian Miers
Computer Science Department – Santa Catarina State University – Joinville, Brazil
E-mail: charles.miers@udesc.br

Adriano Fiorese
Graduate Program in Applied Computing – Santa Catarina State University – Joinville, Brazil
E-mail: adriano.fiorese@udesc.br

Marcos Dias de Assunção
INRIA Avalon, LIP Laboratory, École Normale Supérieure de Lyon – France
E-mail: assuncao@acm.org

Guilherme Piegas Koslovski
Graduate Program in Applied Computing – Santa Catarina State University – Joinville, Brazil
E-mail: guilherme.koslovski@udesc.br

Keywords virtual infrastructure · allocation · data center · SDN · IaaS

1 Introduction

Cloud Computing has revolutionized the provisioning of computing and networking services. Notably, Infrastructure-as-a-Service (IaaS) enables creating Virtual Infrastructures (VIs) [22]; groups of Virtual Machines (VMs) interconnected by virtual networking resources, where the number of resources (*e.g.,* machines, switches, and links) and their configuration (*e.g*, processing and bandwidth) can be dynamically adjusted based on hosted application requirements. Examples of VIs provisioning includes the Virtual Private Clouds (VPCs) offered by public providers (*e.g.*, Amazon VPC, Google VPC), and the private network configuration proposed by OpenStack cloud management framework. The latter enables VI provisioning on small and medium scale data centers.

Network configuration and management are critical tasks in IaaS clouds. VI-hosted applications create large volumes of traffic and spend substantial time performing network activity. For instance, a Facebook cluster can use up to 33% of its running time transferring data [31], a case where under-provisioned virtual networks can drastically affect hosted application performance [36]. The use of network virtualization techniques on data centers aims to improve the performance of legacy applications [16] [8].

By separating the control plane from the data plane, the Software-Defined Networks (SDN) concept introduces opportunities for latency and bandwidth control that can make network management tasks more flexible and hence ease the deployment of VIs. With SDN, a logically centralized controller, aware of the data center network topology and load, is responsible for traffic engineering [20]. In some cases, a VI can have its private controller (or an interface for communication with physical controllers) used for internal management (*e.g.*, load balancing, virtual topology segmentation) [32].

Using SDN to implement virtualized cloud data centers is still challenging as it introduces new dimensions to management such as flow forwarding delay, dynamic virtual topology creation, bandwidth sharing, CPU isolation on switches, and forwarding table sharing [32] [33] [27]. SDN is innovative but can increase the latency as the path that a flow traverses may grow. When proactively configuring a flow along the path, latency increases by a Round-Trip Time (RTT) to the controller. The worst case happens under scenarios that change frequently: controllers interference is eventually required when flow information is periodically removed from the switches' tables [32]. Even when all flow table entries are proactively configured on switches, general purpose SDN-aware virtual switches increase forwarding delay due to hypervisor processing capacity sharing [29].

A cloud management framework uses allocation algorithms to identify and provision physical resources for hosting VIs. Usually, some constraints need to be satisfied during the allocation process to guarantee that the physical infrastructure can provide the requested virtual resources [14]. The problem of allocating cloud data center resources to host VIs is hard due to its computational cost and complexity, and the need to consider a range of constraints from multiple tenants and providers. On IaaS providers, the number of servers composing a data center is a challenging aspect [28], even when pruning the tree of physical candidates by restricting the search to a given data center, region, or zone.

Moreover, the multi-criteria constraints that must be satisfied can exacerbate the VI allocation problem. A VM may require a particular configuration of virtual CPUs, memory, and storage while virtual switches (or even routers) have another set of configuration (*e.g.,* flow table size, memory, and processing power). As a VI extends the IaaS paradigm by including network resources, the allocation problem can be viewed as a virtual network embedding formulation with additional node constraints [7] [14]. The VIs allocation problem, with constraints on virtual resources and topology, belongs to the set of NP-hard problems as other similar problems have already been proven to be in this set or may be reduced to it [6] [14]. The VI allocation is hence as a graph embedding problem: vertices are computing and network equipment whereas edges denote network links and paths. Each graph element has a set of associated requirements or capacities. Virtual graphs carrying out tenant requirements must be placed on a physical graph that represents the cloud data center infrastructure.

In this context, the present work addresses the problem of allocating SDN-based resources for hosting VIs. Moreover, we claim that an allocation model must consider switches' flow tables as a shared resource for provisioning Quality-of-Service (QoS)-aware VIs. Specifically, an SDN-based formulation increases the VI allocation complexity in three main axes: *(i)* New constraints on physical switches capacities are introduced: SDN switches flow tables have a limited size (eventually, newly allocated flows replaced old ones). *(ii)* Flow-table misses: as a virtual resource can be placed anywhere on a virtualized cloud data center, any flow latency is increased by at least one RTT to the controller when a flow-table miss is triggered. *(iii)* Sharing network resources with QoS requirements: while traditional IaaS allocation mechanism enforces a best-effort network sharing, SDN enables the use of network sharing policies increasing the performance of VM-hosted applications.

Existing work provides mechanisms for finding optimal and approximated solutions for VI allocation [14] [5] [9]. Most work allocates only virtual networks and does not tackle computing and switching resources. Some proposals adapt virtual network formulations to SDN-based data centers [23] [35] [12] [18] [11], but do not consider the intricacies of IaaS cloud data centers. In short, this paper makes three main contributions[1]:

- We formulate the online VI allocation in SDN-based cloud data centers as an optimal Mixed Integer Program (MIP). Our formulation considers the main management challenges introduced by SDN, modeling controllers, latency, bandwidth, and switch constraints.
- MIP constraints are relaxed to obtain a linear program, and rounding techniques are applied to propose an acceptable solution. The heuristic innovates by selecting candidates for hosting VIs based on SDN particularities, in addition to VM constraints. Moreover, the proposed mechanism, called QVIA-SDN, explores the data center homogeneity to prune the number of servers and network paths analyzed to decrease the number of comparisons required for processing a VI request.

[1] A previous version of this paper, containing initial results and analysis, was published at CCGrid 2017 [10] and invited to an extended version for this journal.

– Results simulating the application of the proposed mechanism for allocating VI requests atop a cloud data center interconnected by a fat-tree topology [1]. The evaluation quantifies provider-based metrics and tenants perspective by simulating with uniformly distributed loads and VI requests based on commonly used instances on Amazon EC2 provider.

The remainder of this paper is organized as follows. Section 2 discusses related work on VI allocation on SDN-based data centers. Section 3 defines and formulates the VIs allocation problem. Section 4 details the proposed MIPs formulation while Section 5 discusses the proposed heuristic based on relaxing constraints. Experimental results are presented in Section 6, whereas final considerations and future work are discussed in Section 7.

2 Related work

The related work comprehends the allocation of resources to host virtual infrastructures and its provisioning on SDN environments. Table 1 summarizes the literature identifying the main objective, network specification and awareness of SDN particularities.

Table 1 Related work ordered by publication date.

Reference	Network specification	Objective	SDN-aware
Sherwood *et al.* [32]	Network slice, and bandwidth	Network sharing	Yes
Popa *et al.* [30]	Link bandwidth	Bandwidth sharing	No
Chowdhury *et al.* [5]	Switches and bandwidth	Decrease substrate load	No
Drutskoy *et al.* [12]	Topology, controller, address space, and bandwidth	Address space isolation	Yes
Al-Shabibi *et al.* [2]	Topology, address space, and bandwidth	SDN-based platform for network virtualization	Yes
Mijumbi *et al.* [23]	Links, and switches load	Increase acceptance ratio	Yes
Mehmet *et al.* [11]	Switches, links, and controller	Decrease substrate load and minimize controller-to-switch delay	Yes
Feng *et al.* [35]	Bandwidth, and flow table	Proportional allocation of virtual resources	Yes
Oliveira *et al.* [26]	Link bandwidth	Decrease fragmentation	No
Draxler *et al.* [13]	Absent	Decrease response time	No
Bo Li *et al.* [21]	Link bandwidth	Maximize network throughput	Yes

Regarding mechanisms for selecting physical resources for hosting VMs and links, the literature describes optimal formulations and approximation heuristics [14]. In general, each proposal lays down a specific and limited usage scenario, often focusing on optimizing data center metrics, such as fragmentation, cost, revenue, acceptance ratio, among others. For instance, Chowdhury *et al.* [5]

propose a joint allocation of virtual processing and communication resources, as discussed later, whereas Oliveira and colleagues [26] propose a tree-based heuristic to speed up the VIs allocation. The latter represents the network configuration as bandwidth requirements and increases the acceptance ratio without affecting the datacenter fragmentation. However, both mechanisms simplify the network representation by only considering the physical link bandwidth as a potentially shared resource. A parallel work investigated a joint optimization of allocation and reallocation of virtual network services [13]. In this work, we exclusively discuss the allocation problem and leave the discussion on off-line algorithms to reallocate and scale virtual resources as a future work.

The literature defines that virtual network provisioning on IaaS cloud data centers needs to be driven by VM importance (*e.g.,* instance type) or specifically defined (virtual links) [3, 15]. Therefore policies can be applied per physical links (or paths) or considering the data center topology (a global view). For example, a set of policies to approximate network sharing is proposed by Popa *et al.* [30] and can only be achieved in a controlled scenario, as software-defined networks. A parameter indicates to the policies that communication has the same configuration and importance throughout the data center network paths. In our proposal, this parameter is represented by the virtual link abstraction between VMs. Latency and bandwidth requirements are specified for each virtual link.

The allocation of physical resources to host VIs on SDN-based data centers is an open challenge for cloud providers. Initially, Sherwood *et al.* [32] identified the key challenges related to SDN-based virtual resource provisioning, pointing out the importance of flow table and controller sharing among hosted VIs. They propose a hypervisor for instantiating virtual networks, with similar management approaches to those implemented by VMs hypervisors. In addition, Al-Shabibi *et al.* propose a framework for SDN-assisted network virtualization [2]. We propose a QoS-aware allocation algorithm that takes into account the particularities and challenges that the authors [2] have identified while filling out a provisioning gap identified in both works [32] [2].

SDN controller placement was studied in [11]. Authors identified that latency to a controller is a key factor for application performance. A complementary work [12] focused on virtual controller provisioning, highlighting that hosted virtual networks can have distinct addressing schemes and routing policies. The SDN-aware network bandwidth allocation were discussed in [35] [23] [21]. Challenges, features, and objectives individually identified by these works should be combined when allocating data center resources to VIs. In this work, a set constraints addressing the SDN challenges and features are proposed in a MIP formulation to achieve bandwidth and latency control (Section 4). Moreover, we formulate the allocation problem as a joint allocation of VMs, switches, and links, claiming that networking allocation policies can impact the performance of cloud-hosted applications.

3 Problem formulation

To formulate the selection and allocation of physical resources (links, servers, and switches) for hosting VIs, we decompose the problem into *(i)* cloud data center and VI requests representation; *(ii)* resources mapping; and *(iii)* cloud provider objectives.

3.1 Cloud data center and VI requests

Formally, weighted undirected graphs are used to represent cloud data center and VI requests. The former is represented by a graph $G^s(N^s_h, N^s_n, C^s, E^s)$, in which N^s_h is the set of servers and N^s_n the set of switches and routers that compose the physical topology. SDN controllers are denoted by C^s. Links interconnecting servers and network resources are denoted by the set E^s. Each resource (server, switch or link) has a residual (available) capacity denoted by $R(.)$. Similarly, a VI request is denoted by a graph $G^v = (N^v_h, N^v_n, E^v, D^v)$, in which N^v_h is the set of VMs, N^v_n the set of virtual switches, and E^v virtual links. A matrix of maximum allowed latency between virtual endpoints is given by D^v, in which d_{ij} represents the maximum latency between virtual resources i and j. Virtual resources have a capacity requirement represented by c. Table 2 resumes the notation.

Table 2 Notation used along this paper. i and j represent virtual resources while u and v are used for physical ones.

Notation	**Description**
$G^s(N^s_h, N^s_n, C^s, E^s)$	cloud data center graph
N^s_h	data center servers
N^s_n	SDN switches and routers
C^s	SDN controllers
E^s	physical links
$R(.)$	residual capacity of physical resources
$G^v = (N^v_h, N^v_n, E^v, D^v)$	VI request
N^v_h	VMs
N^v_n	virtual SDN switches
E^v	virtual links
D^v	matrix of maximum allowed latencies
d_{ij}	allowed latency between virtual resources i and j
c_i	capacity requirements of virtual resources i
$\Omega(i)$	physical candidates for hosting i
$P^s(i, j)$	set of paths between $\Omega(i)$ and $\Omega(j)$

Cloud providers are moving towards VMs provisioning with QoS network requirements to improve the performance of VI-hosted applications. In this context, VI requests can be designed based on IaaS and Network-as-a-Service (NaaS) scenarios as depicted on Figure 1.

A brief explanation of the problem based on Figure 1 scenario:

- *IaaS-only requests* represent the traditional cloud requests. Tenants request VMs without specifying the details on network topology between virtualized resources. We argue that instead of just specifying the data center, region or zone locations, in short time, a tenant may be able to ask for a maximum latency as well as a minimum end-to-end bandwidth between two VMs. This request is exemplified by the *VI 2* (Fig. 1), in which two VMs must be provisioned with best-effort network configuration. It is worthwhile to note that

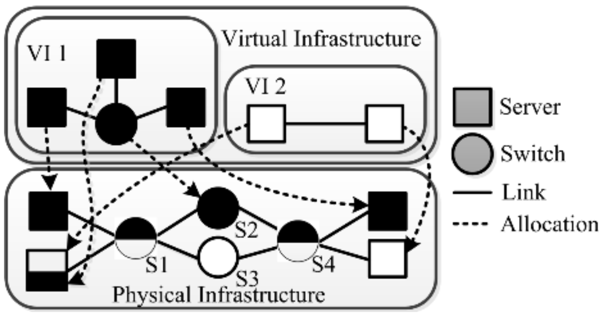


Fig. 1 *E.g.,* VI requests allocation atop an SDN-based cloud provider.

details on data center networking configuration is an undisclosed provider information, and consequently the switches and links path used to enable the VMs communication is abstracted from tenants.

– *Combined IaaS / NaaS requests* enable a full description of VI resources. A tenant details all VMs and intermediate virtual switches, defining the QoS requirement for each composing resource. As exemplified by *VI 1* request (Fig. 1), multiple data center resources may be selected to host a virtual one: the VI requested just one switch, while 3 switches were effectively allocated to deliver the VI. From the tenant's perspective, only the required switch is exposed to a tenant with SDN management capabilities [32].

3.2 Allocating physical resources to host VIs

A cloud orchestration framework processes each VI request, determining whether to accept it or not. In fact, a VI request may ask for the initial provisioning or for elastic reconfiguration of any virtual resource. We argue that initial provisioning and elastic reconfiguration must be independently analyzed and deployed, and consequently the VI reconfiguration is not discussed in the present work. Finally, following the policy used by public cloud providers, the virtual resources are released once the tenant terminates the VI.

The literature defines that the allocation of VI requests onto a cloud data center can be decomposed into nodes and links assignments [37] [5]. A map of VMs onto physical servers is given by $M_h : N_h^v \mapsto N_h^s$, while for virtual switches on SDN-based equipments is denoted as $M_n : N_n^v \mapsto N_n^s$, with $M_h(i) \in N_h^s$ and $M_n(j) \in N_n^s$. Similarly, the map of a virtual link ij is realized onto a physical path $p \in P^s$ between the physical resources that host the end virtual nodes of ij. In other words, $M_e : E^v \mapsto P^s$, $\forall ij \in E^v$ and $M_e(ij) \subseteq P^s(M_h(i) \cup M_n(i), M_h(j) \cup M_n(j))$.

On QoS-aware allocation, the physical resource must have enough capacity to host all virtual resources allocated on it. As the VI requests are received at any time and must be processed on-the-fly, the notation of residual capacity is used to simplify the processing method. In short, the residual capacity represents the remaining available capacity on physical data center resources at a given time. Formally, the capacity constraints for VMs and switches are represented by $c_i \leq R(M_h(i))$ and $c_j \leq R(M_n(j))$, respectively, while for virtual links are given by $c_{ij} \leq R(M_e(ij))$.

3.3 Cloud provider objectives

The first objective in this paper is to allocation multiple VIs delivering the QoS requirements, representing the cloud provider's perspective. Specifically, the main objective is decomposed in decreasing the allocation cost while simultaneously increasing the provider revenue.

In this sense, Equation (1) defines the cost for hosting a VI ($|ij|$ represents the length of path hosting ij in terms of hops). To simplify the formulation, the cost is calculated proportionally to physical capacity reserved [9] [5]. Complementary, the revenue for hosting a VI is given by Eq. (2). Following the same simplification principle, the revenue is related with the QoS configuration delivered to tenants. The abstraction and normalization of cost and revenue equations allow a comparative analysis as presented in Section 6.

$$\mathcal{C}(G^v) = \sum_{i \in N_h^v} c_i + \sum_{j \in N_n^v} c_j + \sum_{ij \in E^v} c_{ij}|ij| \tag{1}$$

$$\mathcal{R}(G^v) = \sum_{i \in N_h^v} c_i + \sum_{j \in N_n^v} c_j + \sum_{ij \in E^v} c_{ij} \tag{2}$$

Finally, the VI provisioning quality, related with SDN configuration, is quantified by the mean latency between communicating virtual resources. The Equation (3) calculates the quality considering all virtual links, where $\mathcal{L}(i,j)$ denotes the mean latency of physical path hosting the virtual link ij, and $|E^v|$ indicates the number of virtual links.

$$\mathcal{Q}(G^v) = \frac{\sum_{ij \in E^v} \mathcal{L}(i,j)}{|E^v|} \tag{3}$$

4 Optimal MIP for QoS-aware VI allocation

4.1 Selecting candidates to host virtual resources

The details on data centers topology and resources are not revealed from providers to tenants. However tenants can select the region or zone for VM instantiation[2], performing an approximation for VM allocation based on geographical data [19]. In our model, the set $\Omega(i)$ represents all physical candidates for hosting a virtual resource i (VM or switch). Following the allocation policy used by public providers, each VM has a zone or region requirement represented by $loc(i)$. In this sense, only the physical servers geographically placed on location $loc(i)$ are candidates for hosting i.

In addition to geographical location (zone and region) selection traditionally used on allocation mechanisms [24] [5], we claim that tenants must detail real network requirements to improve the performance of applications. Indeed, latency and

[2] Amazon EC2 (`https://aws.amazon.com/ec2/pricing/on-demand/`) and Google Compute Engine (`https://cloud.google.com/compute/pricing`) have different prices based on zones/regions.

bandwidth configuration are critical factors to network-intensive applications [34], and are not completely dependent on physical location [28]. In this regard, we propose a candidate selection based on end-to-end latency.

Candidates for hosting virtual switches with latency requirements. The latency requirements can be specified for any pair of virtual resources, however distinct approaches are used to select the physical candidates. When a virtual link i specifies latency requirements for links from/to virtual switches, only physical switches having output links capable of hosting the worst case latency requirement of i are selected. This approach is formally represented by $\Omega(i) = \{u \in N_n^s | max(lat(u,v)) < max(d_{ij})\}; \forall v \in adj(u); \forall j \in adj(i)$, in which $adj(.)$ informs all adjacent resources, and $lat(u,v)$ indicates the latency on physical path u to v.

Candidates for hosting virtual switches without latency requirements. Physical switches with enough residual capacity are candidates for hosting a virtual switch i: $\Omega(i) = \{u \in N_n^s | R(u) \geq c_i\}$.

Candidates for hosting VMs without latency requirements. Physical resources with enough residual capacity are candidates for hosting a VM i. In other words, $\Omega(i) = \{u \in N_h^s | R(u) \geq c_i\}$.

Candidates for hosting VMs with latency requirements. *(a)* VMs connected to virtual switches: in this case, physical candidates are selected based on their latency to virtual switches communicating with VM i. Hence, $\Omega(i)$ is composed of $\{u \in N_h^s | lat(u,v) \leq d_{ij}\}; \forall v \in \Omega(j); j \in adj(i) \setminus N_h^v$.

(b) VMs connected to VMs: in this case, candidates are selected based on end-to-end latency requirement. Thus, $\Omega(i) = \{u \in N_h^s | lat(u,v) \leq d_{ij}\} \forall v \in N_h^s$.

The joint allocation of VMs, switches and virtual links requires a combined analysis of QoS requirements. Formally, the edges of a graph must be mapped on-the-fly with vertices. In this sense, following the specialized literature [5], each virtual resource i is assigned to the physical graph, and interconnected with their candidates through a temporary link with infinite capacity and no communication latency. The core idea behind this approach is to solve a network flow problem simultaneously with vertices allocation. The resulting graph is given by $G^{s'}(N^{s'}, C^s, E^{s'})$, with $N^{s'} = N_h^s \cup N_h^v \cup N_n^s \cup N_n^v$; and $E^{s'} = E^s \cup \{iu \,|\, i \in N_h^v, u \in \Omega(i)\} \cup \{ju \,|\, j \in N_n^v, u \in \Omega(j)\}$. It is worthwhile to mentation that SDN controllers remains part of the augmented graph, as exemplified by Figure 2. The example demonstrate an augmented graph by connecting *VI 1* request from Figure 1 to physical resources, following the previously described approach.

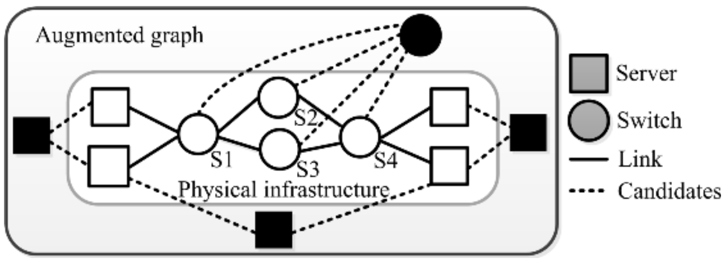


Fig. 2 An augmented graph combining virtual and physical resources.

4.2 Variables and objective

The allocation of SDN-based data center resources to host a VI is identified by a combination of three variables. Specifically, two variables represent the network dimension of the allocation problem (latency and switches flow-tables), while the remaining one is used to indicate map of VMs and virtual switches. From the network flow problem formulation, the variable f_{ijuv} accounts the amount of ij flows allocated on physical link uv. In addition, the binary variable x_{uv} indicates the presence of a virtual link atop u to v. It is set to 1 if $\sum_{ij \in E^v}(f_{ijuv} + f_{ijvu}) > 0$, otherwise 0.

Further to identifying a suitable physical path to host a virtual link, it is necessary to control the switch flow-tables and the SDN controller's usage. The SDN switches may have ephemeral flow-table entries, that is, the forwarding tuple related to the virtual link may be temporally allocated in the controller. In this sense, a binary variable e_{ijpu} indicates the presence of ij flow on the SDN controller (set to 1), or that flow-entry to the path p is hosted on switch u (set to 0).

To achieve the providers and tenants objectives (Section 3.3) our MIP uses a modified version of Equations (1), (2), and (3) to compose the objective function (Equation (4)). Each term of the objective function addresses a map variable (x, e, and f), and the minimization aims at balancing the load atop the data center [5] as well as to decrease the allocation cost for hosting a VI. The load balancing is achieved by giving preference to less loaded resources (by dividing the allocation cost with the residual capacity). The parameters α_{uv}, β_u, and γ_e ($[1, R(.)]$) control the importance level of a resource, while $\delta > 0$ is used to avoid division by zero.

$$\begin{aligned} min : \sum_{u \in N_h^s \cup N_n^s} \frac{\beta_u}{R(u) + \delta} \sum_{i \in N_h^v \cup N_n^v} x_{iu} c_i + \\ \sum_{u \in N_n^s} \frac{\gamma_e}{R(u) + \delta} \sum_{ij \in E^v} \sum_{p \in P^s(i,j)} (1 - e_{ijpu}) + \\ \sum_{uv \in E^s} \frac{\alpha_{uv}}{R(uv) + \delta} \sum_{ij \in E^v} f_{ijuv} \end{aligned} \quad (4)$$

4.3 Constraints

For guaranteeing the QoS of VMs, switches and virtual links, a set of constraints are presented and must be satisfied by the cloud management framework.

Links and servers capacity constraints. The Equation (5) defines capacity constraints for physical links while Equation (6) applies to physical servers. In short, Equation (5) indicates that the sum of all virtual flows passing through the physical link must be less than the residual capacity. The same rationale is applied to physical servers.

$$\sum_{ij \in E^v} (f_{ijuv} + f_{ijvu}) \leq R(uv) x_{uv} \qquad \forall u, v \in N^{s'} \quad (5)$$

$$R(u) \geq \sum_{i \in N_h^v} x_{iu} c_i \qquad \forall u \in N_h^s \quad (6)$$

SDN-related constraints. The limited number of flow-table entries of SDN switches induces a specificity on packet forwarding: virtual flows may pass through switches even when the tuple entry is not present in the local forwarding table. In other words, the variable x may indicate that a switch is part of a forwarding path while the variable e expresses that the corresponding flow-table is present on SDN controller. In this sense, Equation (7) indicates that to calculate the residual capacity of SDN switches the flow-table entries allocated on the SDN controller should not be considered. Specifically, s_{ij} and t_{ij} represent the source and target nodes (VMs or virtual switches) of a virtual link ij.

$$\begin{aligned} R(u) \geq \sum_{k \in N_n^v} \sum_{ij \in E^v : s_{ij}=k \vee t_{ij}=k} \sum_{p \in P^s(i,j)} (x_{ku} - e_{ijpu}) c_k \\ + \sum_{ij \in E^v} \sum_{p \in P^s(i,j)} (x_{iu} - e_{ijpu}) \quad \forall u \in N_n^s \end{aligned} \tag{7}$$

Flow-related constraints. The network details of VI-hosted applications are specified as QoS attributes for virtual links. Regarding the bandwidth requirement, Equations (8) and (9) guarantee that for each virtual link ij, all flows from s_{ij} to t_{ij} must be equals to the virtual link request. Moreover, as exemplified by Figure 2, a virtual link may be allocated atop multiple physical links. In this sense, Equation (10) guarantees that all physical resources from the SDN topology must forward the virtual link requests to the next resource on the hosting path.

$$\sum_{u \in N^{s'}} f_{ijs_{ij}u} - \sum_{u \in N^{s'}} f_{ijus_{ij}} = c_{ij} \quad \forall ij \in E^v \tag{8}$$

$$\sum_{u \in N^{s'}} f_{ijt_{ij}u} - \sum_{u \in N^{s'}} f_{ijut_{ij}} = -c_{ij} \quad \forall ij \in E^v \tag{9}$$

$$\begin{aligned} \sum_{v \in N^{s'}} f_{ijuv} - \sum_{v \in N^{s'}} f_{ijvu} = 0 \\ \forall ij \in E^v, \forall u \in N^{s'} \setminus \{s_{ij}, t_{ij}\} \end{aligned} \tag{10}$$

Latency constraints. In addition to bandwidth requirements, a tenant may specify the maximum tolerated latency between any pair of virtual resources (represented by d_{ij}, for a virtual link ij). However, SDN introduces a management challenge for latency constraints: the flow-table entries may be stored only at the SDN controller, and for each new packet a query to controller framework may be required. In this sense, Equations (11) and (12) guarantee the QoS latency requirement even when the flow-table entry is stored at the controller. In short, the total latency of a physical path (even if the controller is consulted, denotes by variable e) must respect the virtual link requirement.

$$\begin{aligned} d_{ij} \leq \sum_{u,v \in p} (lat(u,v) x_{uv} + lat(u,c) e_{ijpu}) \\ \forall ij \in E^v; \forall p \in P^s(i,j) \end{aligned} \tag{11}$$

$$d_{ij} \geq \sum_{u,v \in p} lat(u,v) x_{uv} \quad \forall ij \in E^v; \forall p \in P^s(i,j) \tag{12}$$

Meta and binary constraints. A set of meta and binary constraints are defined to ensure cohesion of VMs, switches, and links mapping. Initially, a virtual resource (VM or switch) must be allocated by a single data center resource as given by Equation (13). In turn, Equation (14) indicates that forward and reverse flows must be allocated on the same path. Finally, Equations (15) to (17) ensure the domains for all variables.

$$\sum_{u \in \Omega(i)} x_{iu} = 1 \qquad \forall i \in N_h^v \cup N_n^v \tag{13}$$

$$x_{uv} = x_{vu} \qquad \forall u, v \in N^{s'} \tag{14}$$

$$f_{ijuv} \geq 0 \qquad \forall u, v \in N^{s'}; \forall ij \in E^v \tag{15}$$

$$x_{uv} \in \{0, 1\} \qquad \forall u, v \in N^{s'} \tag{16}$$

$$e_{ijpu} \in \{0, 1\} \qquad \forall u \in N_n^s; \forall ij \in E^v; \forall p \in P^s(i, j) \tag{17}$$

5 QVIA-SDN mechanism

A MIP-based modelling is efficient to identify and formalize the problem, however solving a MIP is known to be computationally intractable [4] [5]. This statement poses a barrier in the use of the optimal MIP for QoS-aware VI allocation in real cloud data center scenarios. Faced with this fact, a set of techniques are applied to relax the integer constraints obtaining an Linear Program (LP) and to decrease the search space (the number of physical candidates and data center paths). The techniques (relaxed LP, search space pruning, and rounding algorithms) compose QVIA-SDN (QoS-Aware VI Allocation on SDN-based data centers), and are sequently executed by the cloud management framework. First, the LP is solved and the resulting approximated solution is latter treated by a rounding heuristic.

5.1 Relaxing MIP variables

Initially, the variables domains defined by Equations (16) and (17) are relaxed originating Equations (18) and (19). It is worthwhile to highlight that all other constraints and the objective function remain unchanged.

$$1 \geq x_{uv} \geq 0 \qquad \forall u, v \in N^{s'} \tag{18}$$

$$1 \geq e_{ijpu} \geq 0 \qquad \forall u \in N_n^s; \forall ij \in E^v; \forall p \in P^s(i, j) \tag{19}$$

5.2 Pruning cloud data center candidates

Even relaxing the domain of variables x and e, the dimensionality of the allocation problem remains an obstacle. On the one hand thousands of servers can be combined to compose cloud data centers, while on the other hand a tenant may request a VI composed of hundreds of virtual resources. This scenario can be mitigated by using knowledge on the data center composition: usually cloud data centers are composed by homogeneous servers interconnected by structured network topologies [34]. Independently of the network topology, cloud data center servers may be grouped by similar aspects (*e.g.,* processing capacity, bandwidth and latency) composing groups of candidates [31]. In this sense, QVIA-SDN reduces the search space by enforcing a maximum percentage of physical resources analyzed per data center region. The percentage per region is adjustable and the impact on allocation quality is discussed in Section 6.

5.3 Decreasing the number of physical paths

On modern data centers topologies, multiple paths between servers are available to improve fault-tolerance, reliability, and enable load balancing [34, 1]. Although effective for the aforementioned objectives, accounting and controlling all paths between servers and switches is practically infeasible for online VI allocation. However, the MIP model requires the set P^s (representing all physical paths available to host the VI request) to formulate the allocation model. For speeding up the allocation, QVIA-SDN accounts just a subset of physical paths for composing P^s. It is worthwhile to highlight that this temporary hiding of physical paths does not compromise the reliability as any algorithm or backup-traffic engineering can be later applied. In summary, P^s is composed of *(i)* one shortest path and *(ii)* one small latency path between all pairs of physical candidates (servers and switches), and *(i)* is different from *(ii)*.

5.4 Rounding heuristic

The joint execution of the relaxed LP with techniques to prune physical candidates (servers and links) soften the dimensionality barrier of the allocation. However, the values obtained for variables x and e are no longer binary and the correlation between f, x and e is lost. In order to guarantee the QoS and integrity of VI allocation, QVIA-SDN employs a rounding heuristic to evaluate the variable values. In short, two steps are performed by QVIA-SDN, as given by Algorithms 1 and 2.

The Algorithm 1, based on D-VINE proposal [5], implements the techniques to select the candidates for hosting VMs and switches, as discussed in Section 4.1. Initially, an augmented graph is created to connect the virtual resources to their respective physical candidates. Latter, the augmented graph is given as input to solve the LP (lines 1 and 2). As previously discussed, after solving the LP the variables x and e offer approximated values. In this sense, from lines 3 to 16 of Alg. 1 a suitable hosting candidate is identified for each virtual resource. The rationale of this heuristic is to reconstruct the correlation between variables x and f. Formally, p_z denotes a weighted product of x_{kz} and the total flow passing through kz, where k represents a virtual resource: $p_z = \alpha(\sum_{ij \in E^v} f_{ijkz} + f_{ijzk}) + (1 - \alpha)x_{kz}$. Although QVIA-SDN performs a joint allocation of VMs, switches, and virtual links, the cloud administrator can guide the heuristic preference by setting the weight α.

A VI allocation is rejected when no candidates are identified for hosting any virtual resource (lines 8 to 9). When multiple physical candidates are identified, the candidate with the highest p_z is selected. After identifying a suitable mapping for each virtual resource, the virtual network interconnection must be reserved. At this point, QVIA-SDN executes the Deterministic Path Search (DPS) algorithm

```
   Input: $G^v, G^s$
   Output: $M_n, M_h, M_e$
1  Create augmented graph $G^{s'}$
2  Solve QVIA-SDN with relaxed variables
3  for $k \in N_h^v \cup N_n^v$ do
4      for $z \in \Omega(k)$ do
5          $p_z = \alpha(\sum_{ij \in E^v} f_{ijkz} + f_{ijzk}) + (1-\alpha)x_{kz}$
6      end
7      Let $z_{max} = \max\{p_z | z \in \Omega(k)\}$
8      if  $z_{max} = \emptyset$ then
9          Reject $G^v$
10     end
11     if $k \in N_h^v$ then
12         Set $M_h(k) \leftarrow z_{max}$
13     else
14         Set $M_n(k) \leftarrow z_{max}$
15     end
16 end
17 if $DPS(G^v, G^{s'}, M_n, M_h, M_e)$ then
18     Update residual capacities of physical resources
19     Return $M_n, M_h, M_e$
20 else
21     Reject $G^v$
22 end
```

Algorithm 1: Pseudocode for QVIA-SDN, adapted from [5].

(described in Alg. 2) to analyse data center network topology considering the SDN particularities regarding controllers and switches. For a given virtual link ij, DPS analyzes bandwidth and latency requirements as well as values obtained for variable e. The rationale of DPS is to identify whether a virtual link can be hosted by a physical path decreasing the hops and flow-table entries, when possible.

Independently of the virtualization technique applied on the cloud data center servers, when two VMs are allocated on a single server it is assumed that the server can provide the bandwidth and latency requirements (lines 2 to 4). At lines 5 to 14 of Algorithm 2, all physical paths candidates to host a virtual link ij are analyzed considering the possible allocation of flow-table entries at the SDN controller, identified by the variable e. Physical paths with latency higher than d_{ij} or without the bandwidth requirement are discarded. It is worthwhile to observe that in some cases a physical path identified as inappropriate can be reconfigured to attend the latency configuration. In other words, a knapsack problem involving all resources along the physical path must be solved to accommodate the virtual link request. When multiple paths are identified as potential candidates (respecting latency and bandwidth requirements), we selected those that minimize the hops and switch flow-table entries. Finally, a VI is rejected when a virtual link can not be accommodated.

6 Simulations and results

The evaluation quantifies the data center usage (from a provider's perspective), as well as the QoS delivered to tenants. Initially, the metrics are described in Section (6.1) while the simulation details are given in Sections (6.2) and (6.3). QVIA-SDN and a discrete event simulator were implemented in Java v1.8, and for solving the LP the CPLEX (v12.6.1.0) framework was used. All experiments

```
Input: $G^v, G^{s'}, M_n, M_h, M_e$
Output: True or false and paths for $M_e$
for $ij \in E^v$ do
    if $M_n(i) == M_n(j) \vee M_h(i) == M_h(j)$ then
        continue
    end
    for $p \in P^s(i, j)$ do
        $path \leftarrow \{\}$
        $lat\_path \leftarrow 0$
        for $u \in p$ do
            if $e_{ijpu} > 0$ then
                $lat\_path \leftarrow lat\_path + lat(u, c)$ $path \leftarrow path + u + controller(u)$
            else
                $path \leftarrow path + u$
            end
        end
        if $R(path) \geq c_{ij} \wedge lat\_path \leq d_{ij}$ then
            Set $M_e(ij) \leftarrow path$
            break
        else
            $path = solveKnapsack(path, d_{ij})$
            if $path$ then
                Set $M_e(ij) \leftarrow path$
                break
            else
                Reject $G^v$
            end
        end
    end
end
```

Algorithm 2: Pseudocode for Deterministic Path Search.

were performed on Intel Xeon E5-2620 2.0GHz/24 cores, 196GB/RAM. Finally, simulation results comparing QVIA-SDN with baseline mechanisms are discussed in Sections (6.4), (6.5), and (6.6).

6.1 Metrics

A set of five metrics were selected to represent the cloud provider and tenants perspectives, as discussed in Section (3.3).

- *Revenue-to-cost ratio*: by calculating the relationship between revenue, Equation (2), and cost, Equation (1), the provider obtains a vision on the volume of resources invested to accept a VI request.
- Cloud data center *fragmentation*: as cloud data centers are composed of thousands of servers, to reduce the number of active resources is an intuitive economical objective. In this sense, the fragmentation of physical servers and links is calculated by dividing the number of active resources by the total number of available resources.
- Although the allocation of VI with constraints on edges and vertices belongs to NP-hard, the *mean runtime to allocate* a VI request is essencial to verify the applicability of QVIA-SDN on real cloud scenarios.
- Combined with the revenue-to-cost ratio and data center fragmentation, the *acceptance ratio* of VIs gives a complete understanding on the cloud provider's perspective. In addition, we accounted the reasons for rejection of VIs.

– Finally, the tenant's perspective is represented by the *mean latency* of a VI, as defined by Equation (3).

6.2 Cloud datacenter topology

To analyze the QVIA-SDN applicability on cloud scenarios, the fat-tree topology was selected to represent the cloud data center [1] [25] [34]. A fat-tree is based on k-ports switches and organized in pods. Each pod contains two layers of $k/2$ switches (aggregation and edge). The aggregation switches are interconnected to core switches (there are $(k/2)^2$ k-port core switches). A fat-tree supports $k^3/4$ hosts. Exemplifying a fat-tree, Figure 3 depicts a topology based on $k = 4$.

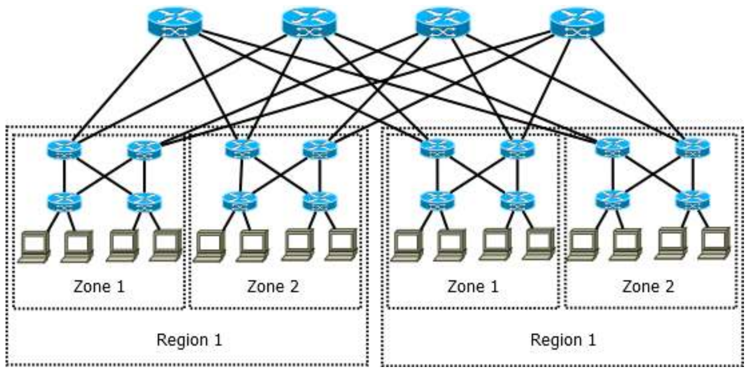


Fig. 3 A hierarchical organization of zones and regions on a data center based on fat-tree topology.

In this paper, we discuss two configurations, in which: $k = 4$ and $k = 8$. Inspired on public cloud providers as Amazon and Google that organize the servers on zones and regions, a hierarchical organization was performed. As given by Figure 3, each pod represents a zone, and a pair of pods denotes a region.

6.3 VIs requests

Aiming to approximate the simulation from real cloud scenarios, two commonly utilized VI topologies on public and private clouds were selected: n-layers (NL) and Virtual Private Cloud (VPC).

– ***n*-layers VI requests.** The arrangement of VMs following a n-layers topology is commonly performed by cloud tenants [17]. In short, the VI is composed of a load balancer used to distributed the end-user requests for multiple VMs. Eventually, queries to database resources (the final layer) are performed by the intermediate layers. For including the network QoS requirements, each layer is interconnected by a virtual switch, while all switches are interconnected by virtual private backbone.

– **VPC requests.** Amazon EC2 and Google Computing Engine introduced the dynamic provision of VPCs[3] composed of a subset of access point rules and a set of VMs attached to it, composing private Local Area Networks (LANs) that are managed by the tenant. For composing VPC requests, a set of VMs is connected to a SDN-based virtual switch.

To analyze the QVIA-SDN performance and applicability, two sets of simulations were performed varying the composition of NL and VPC requests. The first scenario consists of requests based on a uniform distribution of processing and networking configurations, whereas the second one is based on commonly used VM instances on Amazon EC2.

6.4 Simulation with uniformly distributed loads

Representation of the physical capacity of servers, switches, and links are absolute values defined as: 100 for core, aggregation, and edge switches representing the flow-table size; and 1000 for servers (denoting processing, memory, or storage). The bandwidth capacity between core switches and pods is defined as 10 Gbps, and as 1 Gbps for links inside the pod. The latency between any pair of physical resources is defined as 1 ms, while the latency between core switches and the SDN controller is 2 ms. Core switches are directly connected with the SDN controller, while other switches are connected by logical paths.

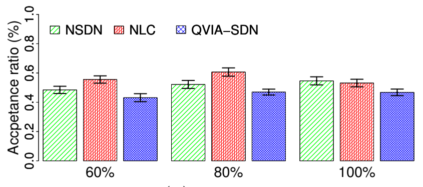
In this scenario, the number of VMs composing VI requests and their configuration were randomly selected. This simulation aims to identify the behavior of the QVIA-SDN, the quality of the solution and the data center stress in front to uniformly distributed loads representing a cloud data center with high variability on VIs configurations. On multi-tiered VI requests, the number of VMs for web servers and databases is uniformly distributed between 10 and 20, while for VPC the number of VMs follows an uniform distribution between 5 and 10 elements. On both VI topologies, the virtual capacity is defined as a fraction of total physical capacity: each virtual switch, VM, or virtual link, consumes $5 - 15\%$ of a physical resource (following a uniform distribution). As cloud data centers are organized in regions and zones (as proposed by Figure 3), the geographical location of VIs is defined by an initial random selection of region, followed by a specification of a zone (randomly performed). In short, all VIs define a region, with 50% chance to get a zone. A set of 50 requests (VPC or NL) is submitted for each physical scenario. Each scenario is simulated during 100 discrete intervals, and VI arriving times are uniformly distributed in this period. A VI remains active for at most 30 intervals.

QVIA-SDN is compared with two baseline algorithms. First, a formulation without SDN knowledge and latency control, labeled as Non-SDN (NSDN), represents a common approach on literature [5]. In order to isolate the QVIA-SDN latency-control overhead, a version without latency optimization constraints is presented and identified by No Latency Control (NLC) label. Each scenario is executed with a limited number of physical candidates identified by the percentage (60, 80 and 100%) to investigate the candidates pruning discussed in Section 5.2. Based

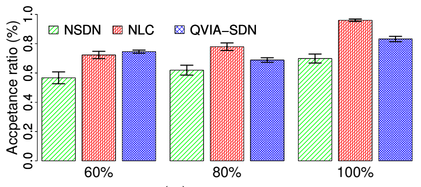
[3] Virtual Private Cloud: `https://aws.amazon.com/vpc` and `https://cloud.google.com/compute/docs/vpc/`.

on empirical observations, QVIA-SDN parametrization is $\alpha = 0.9$ (Section 5.4) and $\beta_u = \gamma_e = \alpha_{uv} = 1$ (Section 4.2). The results show sample means with 95% confidence intervals.

Acceptance ratio. The acceptance ratio for $k = 4$ and $k = 8$ scenarios is presented in Figures 4(a) and 4(b), respectively. Due to limited number of available physical resources on $k = 4$ configuration, Fig. 4(a) indicates a small variation on results, independently of the number of physical candidates (60, 80 or 100%). A different perspective is highlighted by Fig. 4(b): as QVIA-SDN tends to distribute groups of virtual resources atop the substrate (for decreasing the internal average latency), the mechanism can increase the acceptance ratio when more physical resources are considered.



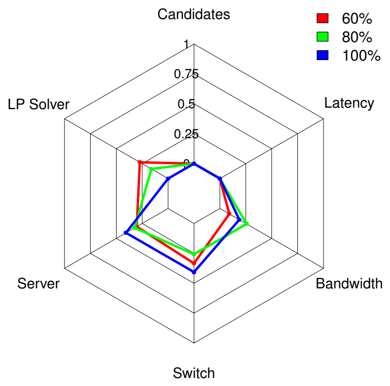
(a) $k = 4$.



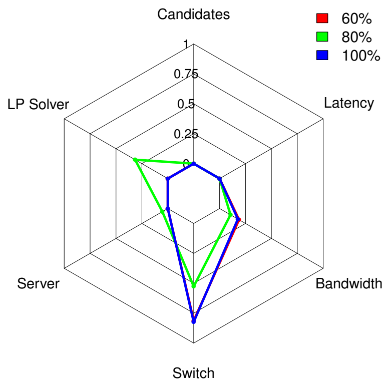
(b) $k = 8$.

Fig. 4 VI requests acceptance ratio.

Figure 5 summarizes the QVIA-SDN rejection reasons for uniformly distributed loads on $k = 4$ and $k = 8$ fat-tree configurations. With a limit number of physical resources ($k = 4$, Fig. 5(a)), the LP solver was unable to find a suitable solution with 60% and 80% pruning, whereas all requests were processed using 100% of available candidates. Independently of the pruning configuration, the servers, switches and network loads impose an allocation limit for this scenario. Results for a $k = 8$ configuration indicate a limit on switches configuration, even when all resources composing a data center are considered by QVIA-SDN.



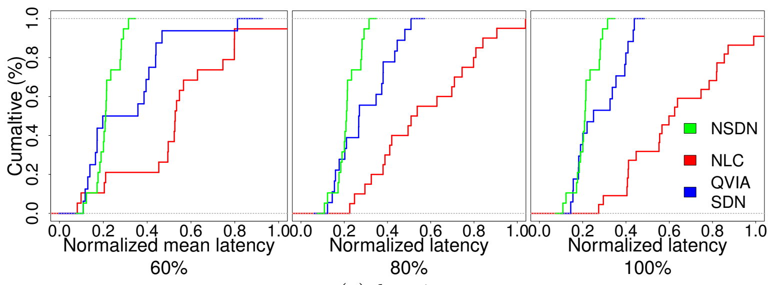
(a) Uniformly distributed loads and $k = 4$.



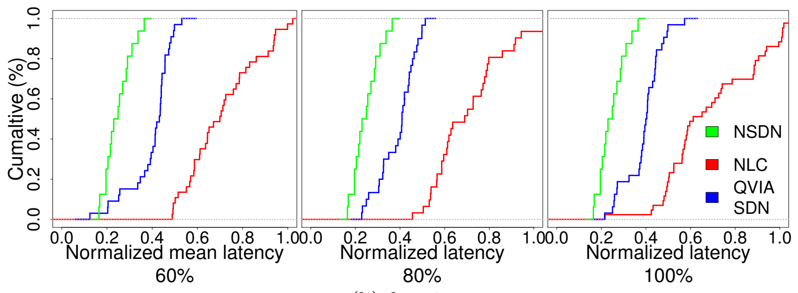
(b) Uniformly distributed loads and $k = 8$.

Fig. 5 Reasons for rejecting VI requests with QVIA-SDN.

Average latency. Figure 6 shows the cumulative distribution of normalized mean latency. The average latency experienced by tenants provisioned with QVIA-SDN allocations is smaller than the latency experienced by allocations performed by the algorithm without latency control (NLC), but in some cases higher than latency in the environment without SDN. This fact is justified by the number of accepted VIs: as NSDN allocates a smaller number of requests, all flow-table entries are placed on switches.



(a) $k = 4$.

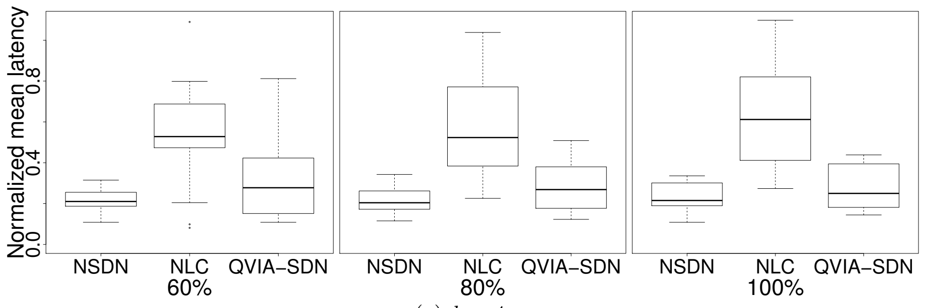


(b) $k = 8$.

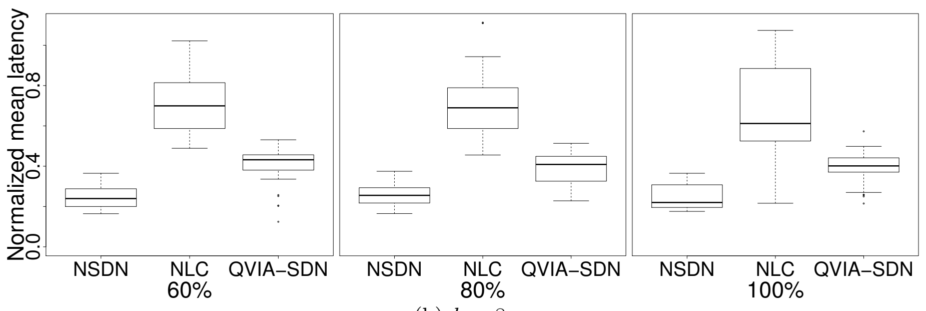
Fig. 6 Cumulative distribution of normalized mean latency.

QVIA-SDN uses the SDN controller to allocate flow-table entries and still respect the latency requirements, justifying the introduction of such latency-aware mechanism. Complementary, Figure 7 indicates that QVIA-SDN presents small variability on latency compared to NLC. As NLC has no latency control mechanism, results show a high variability, which can harm the performance of hosted applications. On its turn, NSDN results on smallest variability on latency as all flow-table entries are allocated on switches.

Data center fragmentation. Figures 8(a) and 8(b) present the fragmentation for $k = 4$ and $k = 8$, respectively. QVIA-SDN tends to condense virtual resources atop the data center. On both scenarios, QVIA-SDN obtained a fragmented switch configuration, due to the high use of switches to guarantee latency requirements. It is worthwhile to highlight that even with a higher acceptance ratio, both SDN-aware versions (NLC and QVIA-SDN) have competitive or better results for data center fragmentation metric face to NSDN algorithm.



(a) $k = 4$.



(b) $k = 8$.

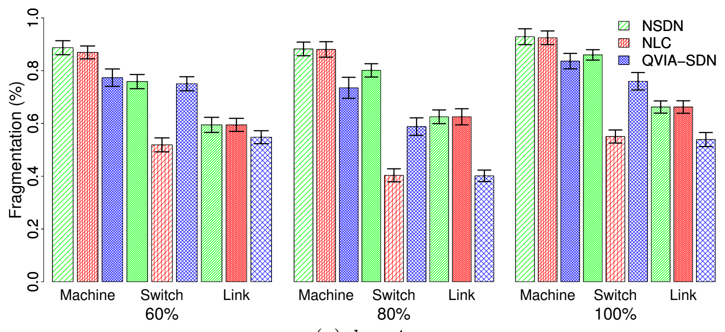
Fig. 7 Mean latency variability.

Revenue-to-cost ratio. A cloud provider aims to allocate more VI requests using a minimum subset of physical resources. In this sense, revenue-to-cost ratio quantifies the proportion of physical resources reserved for hosting a VI. One may expect that as QVIA-SDN performs allocations on the SDN controller and consequently requires switches-to-controller communications, the average cost is increased. However, Figures 9(a) and 9(b) indicate that all three versions have equivalent values. However, it is important to highlight that even allocating more VI requests, the decreased fragmentation induced by QVIA-SDN keeps a revenue-to-cost ratio with low and competitive values. Moreover, QVIA-SDN allocates flow-table entries on SDN controllers decreasing the switches usage.

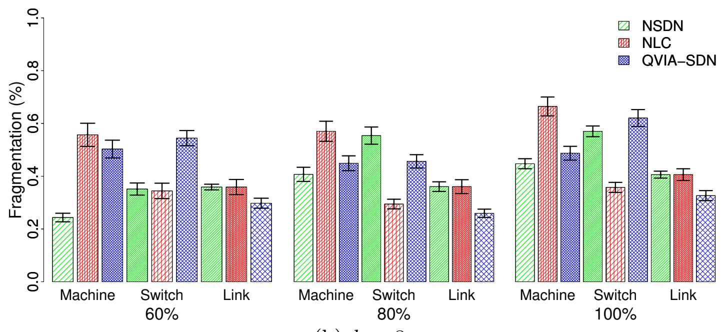
Mean runtime to allocate VI requests. Figures 10(a) and 10(b) show the mean runtime to allocate VIs. As expected, the introduction of SDN requirements increased the number of constraints on LP, and consequently the runtime. As the other two algorithms are more lenient (less restrictive) they are composed of a smaller number of constraints. Despite this fact, the average runtime is in the order of a few seconds for most cases.

6.5 Simulation based on $m3$ instance type from Amazon EC2

Usually, public cloud providers omit details on their internal data center set up including servers and switches configurations. Hence, we model the physical servers based on the hardware configuration of a private cloud that hosted our simulations. The experiments were performed on Intel Xeon E5-2620 2.0GHz - 24 cores, 196GB

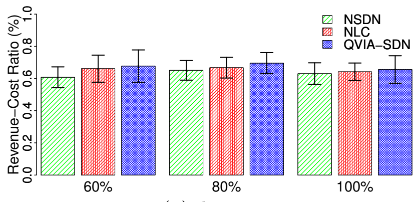


(a) $k = 4$.

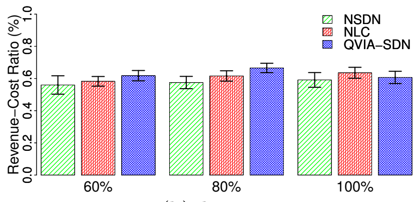


(b) $k = 8$.

Fig. 8 Data center fragmentation.



(a) $k = 4$.

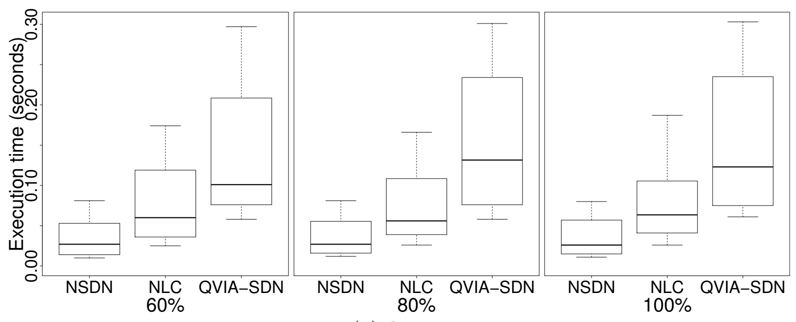


(b) $k = 8$.

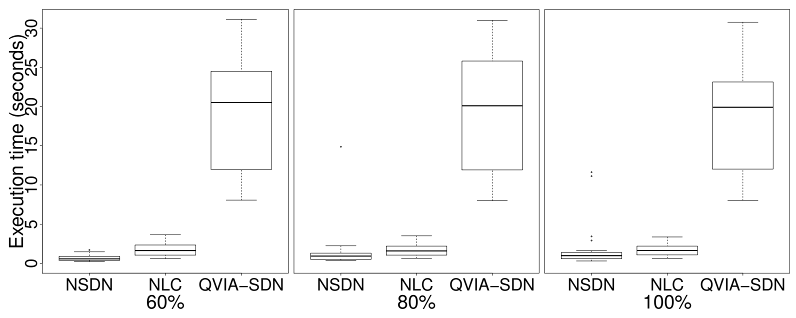
Fig. 9 Revenue-to-cost ratio.

(DDR3) RAM, 2 TB storage. The capacity of each physical server (CPU, RAM, and storage) was represented by a single weighted value. The weights correspond to the resource importance on the allocation process and were empirically defined: 50% for CPU, and 25% for RAM and storage. Based on the real server on which the experimental analysis was performed, the weighted capacity for each data center server is equal to 576. For core, aggregation, and edge switches, the flow-table size is defined as 100 entries, whereas the SDN controller has no limit.

The bandwidth between core switches and pods is 10Gbps while intra-pod links is defined with 1Gbps. The communication latency between physical pair is 1ms while the latency until the SDN controller is 2ms. The core switches are physically connected to the SDN controller whereas the remaining switches communicate



(a) $k = 4$.



(b) $k = 8$.

Fig. 10 Mean runtime to allocate VI requests.

through logical paths, and consequently the total latency increases according to the path length.

In this scenario, the capacity used for composing VI requests is based on commonly used $m3$ VM instances [28] from Amazon EC2[4]. The goal is to identify the QVIA-SDN behavior when allocating popular configurations from public cloud scenarios on SDN-based data centers. Thus, four configurations from $m3$ instance types were selected, *medium*, *large*, *xlarge* and *2xlarge*, summarized in Table 3. The VM requirement is accounted exactly as performed for physical servers capacity, using weights for representing the resource importance, defined as 2, 10, 25 and 51 for *medium*, *large*, *xlarge* and *2xlarge*, respectively. The capacities of virtual links, representing the communication between virtual resources, follow a uniform distribution selecting 10, 20, 40 or 80% of average bandwidth usage per instance type [28] (last column from Table 3). Regarding the location requirement, all VIs have a region assigned to them. Moreover, there is a 50% probability of a VI receiving a specific zone.

A set of 50 requests (VPC or NL) is submitted for each physical scenario. VI arriving times are uniformly distributed up to 100 discrete intervals (a VI remains active for at most 30 intervals). The results show sample means with 95% confidence intervals. Since the objective of this scenario is to identify the behavior of QVIA-SDN on SDN-based scenarios, the mechanism without SDN control, NSDN, is omitted. QVIA-SDN is compared only to No Latency Control (NLC).

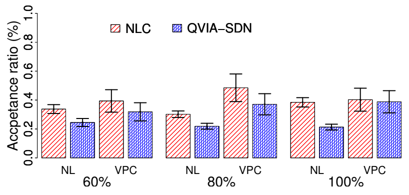
[4] $m3$ VM instance - Amazon EC2: `https://aws.amazon.com/pt/ec2/instance-types/`

Table 3 Configuration of $m3$ instance types from Amazon EC2.

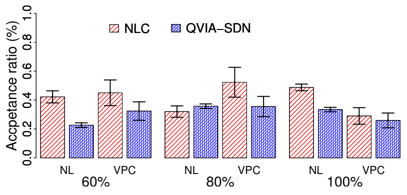
Instance type	**vCPU**	**RAM (GB)**	**Storage (GB)**	**Bandwidth (Mbps)**
medium	1	3.75	4	492
large	2	7.5	32	739
xlarge	4	15	80	958
2xlarge	8	30	160	1188

Over again, each scenario is executed with a limited number of physical candidates identified by the percentage (60, 80 and 100%) to investigate the candidates pruning, and previously applied configuration are retained ($\alpha = 0.9$ and $\beta_u = \gamma_e = \alpha_{uv} = 1$).

Acceptance ratio. Figure 11 summarizes the results for $k = 4$ and $k = 8$ configurations. Specifically, Figure 11(a) indicates that on small scale data centers, the acceptance ratio of VPC requests overcomes the NL one. Additionally, regardless of the number of physical candidates, the acceptance ratio has little variance on results. For $k = 8$ (Figure 11(b)) there is an equivalence between the two types of VI requests.



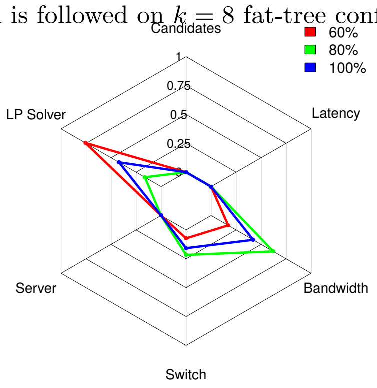
(a) $k = 4$.



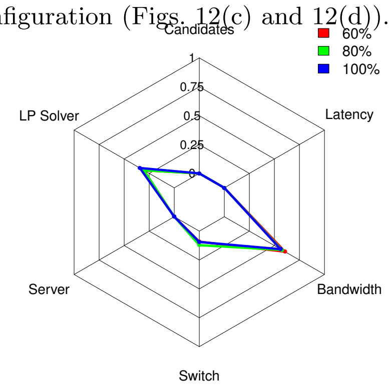
(b) $k = 8$.

Fig. 11 VI requests acceptance ratio.

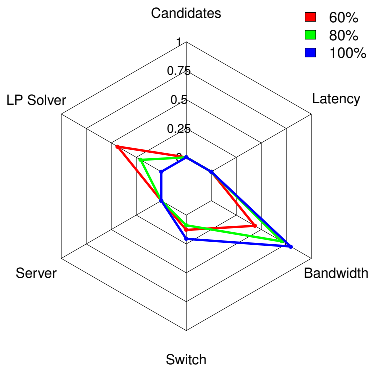
Figure 12 summarizes the QVIA-SDN rejection reasons for VI requests based on $m3$ instance type. Considering a small scale data center (fat-tree with $k = 4$), for NLs and VPCs requests, the main reasons for rejection are insufficient bandwidth and no solution for LP solver, as given by Figs. 12(a) and 12(b), respectively. The pattern is followed on $k = 8$ fat-tree configuration (Figs. 12(c) and 12(d)).



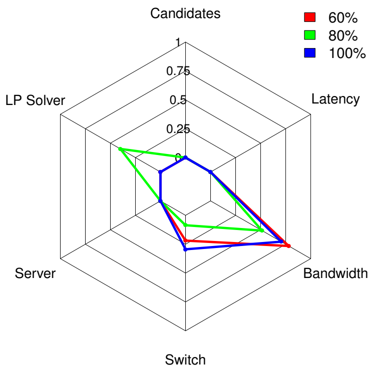
(a) NL and $k = 4$.



(b) VPC and $k = 4$.



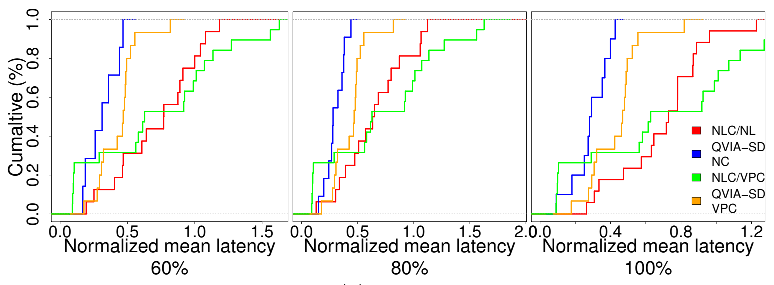
(c) NL and $k = 8$.



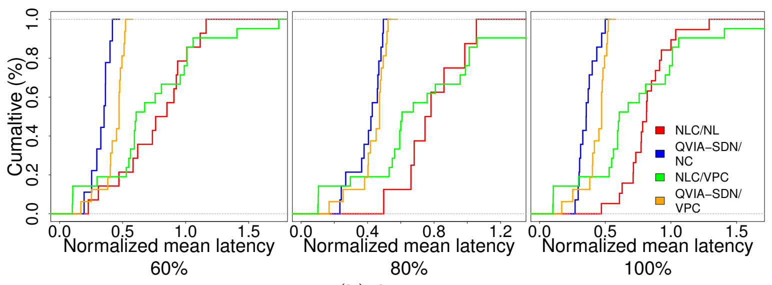
(d) VPC and $k = 8$.

Fig. 12 Reasons for rejecting VI requests based on $m3$ instance type.

Average latency. Figure 13 shows the cumulative distribution of normalized mean latency. The VIs allocated with QVIA-SDN have a lower average latency, independent of the request type, whereas NL requests present the lowest latencies for $k = 4$ and $k = 8$ configurations. Another point worth mentioning is that with $k = 8$ configuration, the latency difference gained evidence as the number of possible hosting paths is increased. NL presented a lower average latency, in other words, it used more flow-table entries in the switches, which resulted in a lower acceptance ratio. Finally, due to the latency control mechanism introduced by SDN, the latency variability was lower on QVIA-SDN allocations (Figure 14).

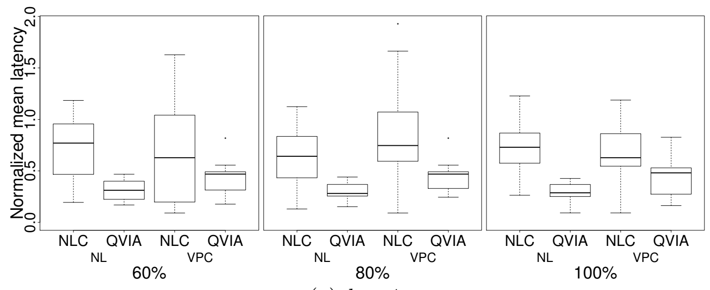


(a) $k = 4$.

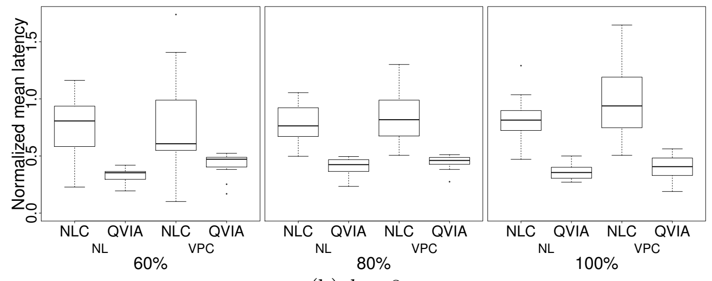


(b) $k = 8$.

Fig. 13 Cumulative distribution of normalized mean latency.



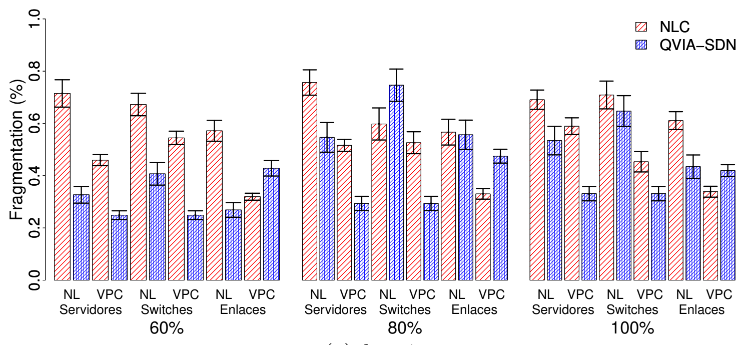
(a) $k = 4$.



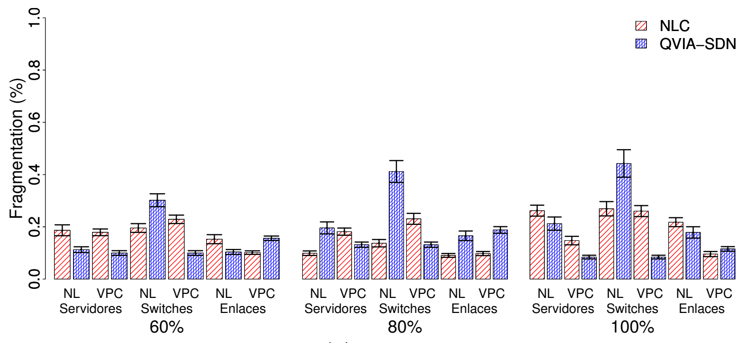
(b) $k = 8$.

Fig. 14 Mean latency variability.

Data center fragmentation. Figures 15(a) and 15(b) summarize the results for $k = 4$ and $k = 8$ configurations, respectively. Fragmentation results follow the average latency observed. Low latency goal acts on the proximity of resources composing a VI. Indeed, since each VI has a different location, the low latency goal (objective function and constraints) does not allow distancing the VI resources to perform the allocation in physical servers that are already allocating other VIs. Thus, the data center fragmentation is increased. As NL obtained a lower average latency, it also had a greater fragmentation as more servers and switches were used for hosting VIs. Consequently, the acceptance ratio was decreased.



(a) $k = 4$.

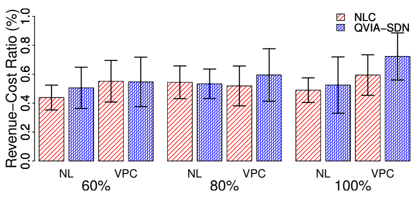


(b) $k = 8$.

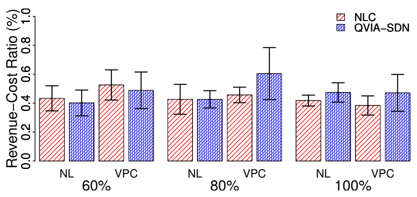
Fig. 15 Data center fragmentation.

Revenue-to-cost ratio. Figures 16(a) and 16(b) indicate an equivalence between QVIA-SDN and NLC mechanisms, as well as NL and VPC requests. It is worthwhile to correlate the revenue-to-cost values with the acceptance ratio. NLC obtained greater acceptance ratio, but a smaller revenue-to-cost ratio. That is, QVIA-SDN allocates VI requests with loads (revenue) greater than those allocated by NLC. The same rationale can be applied to VPC with respect to NL requests. Finally, using QVIA-SDN a provider can improve the latency perspective of hosted VIs without harming the metrics that represent the administrative and economical perspectives.

Mean runtime to allocate VI requests. The runtime of QVIA-SDN and NLC are presented in Figure 17. As previously commented, due of the larger set of constraints, QVIA-SDN requires more running time to analyze a VI request. Indeed, the worst case is observed for NL requests (with complex virtual network topology). However, the allocation time does not exceed 30 seconds in the worst case.

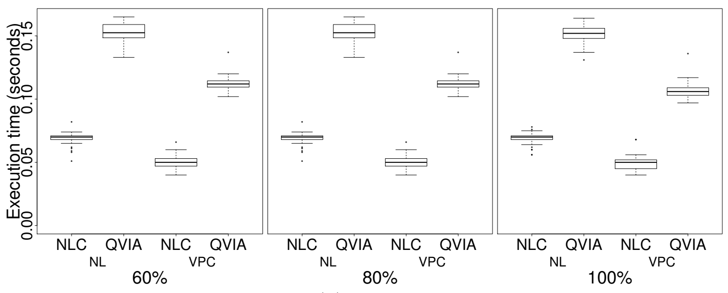


(a) $k = 4$.

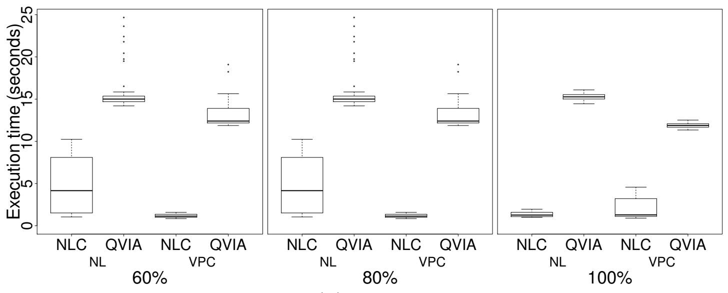


(b) $k = 8$.

Fig. 16 Revenue-to-cost ratio.



(a) $k = 4$.



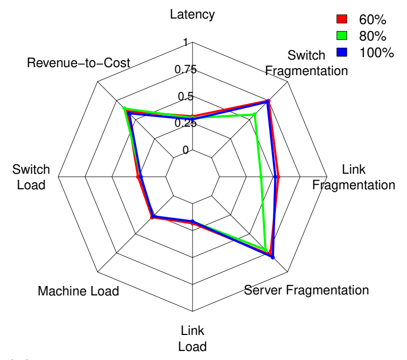
(b) $k = 8$.

Fig. 17 Mean runtime to allocate VI requests.

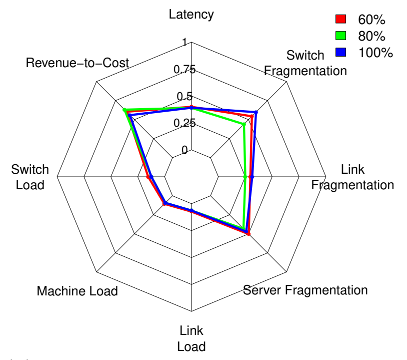
6.6 Key observations

Figure 18 summarizes results for uniformly distributed and $m3$-based loads for some metrics analyzed on simulations. Data is normalized by maximum values for composing the graphs.

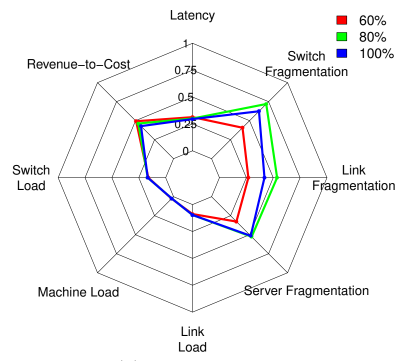
An SDN-aware allocation mechanism can increase the acceptance ratio without decreasing the QoS. In general, the results indicate that it is possible to provide QoS guarantees without affecting the acceptance ratio. Moreover, even allocating more VI requests, QVIA-SDN still composes VIs with lower internal latency values. The first test scenario (Section 6.4) indicated that it is possible to perform the allocation in SDN environments, providing QoS guarantees, and maintaining a competitive acceptance ratio when compared to baseline solutions. The second test set (Section 6.5) evaluated the performance of QVIA-SDN in scenarios close to those experienced by cloud providers. When allocated with QVIA-SDN, the NL requests had a lower mean latency (for all scenarios, as given by Fig. 18), but VPC



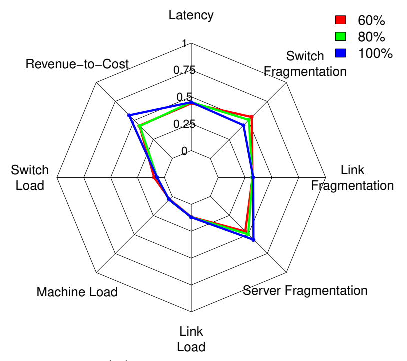
(a) Uniformly distributed load and $k = 4$.



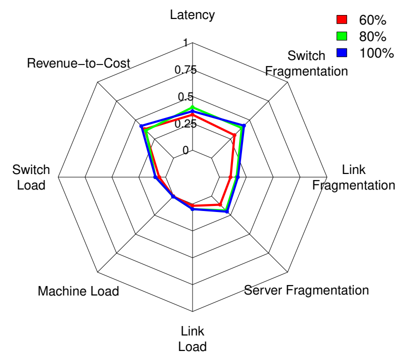
(b) Uniformly distributed load and $k = 8$.



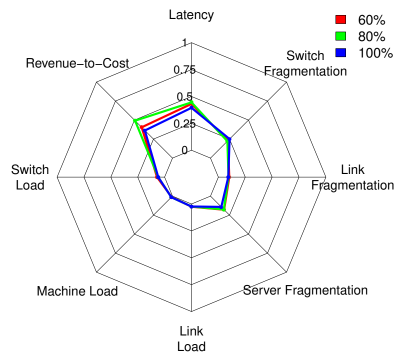
(c) NL and $k = 4$.



(d) VPC and $k = 4$.



(e) NL and $k = 8$.



(f) VPC and $k = 8$.

Fig. 18 Key observations with uniformly distributed and $m3$ based loads.

can be considered the most cost-effective virtual infrastructure for both clients and provider.

QoS requirements can be provisioned without decreasing the revenue-to-cost ratio. The cloud providers objectives can be achieved without overloading the data center as indicated by the fragmentation metric. From the client's point-of-view, QVIA-SDN allocates VPC respecting the QoS requirements but sacrificing the average latency. From the provider's perspective, VPC requests allocated by QVIA-

SDN obtained better results in all aspects (acceptance ratio, fragmentation, and higher revenue-to-cost ratio). In addition, an SDN-aware mechanism can increase the provider revenue. The different percentages of evaluated candidates indicate that it is possible to find a suitable solution without evaluating the entire search space.

Due to NP-hard complexity, runtime is an open challenge. Although QVIA-SDN indicates that QoS-aware provisioning is possible even considering latency requirements on SDN-based data center, the NP-hard complexity remains a barrier. The experimental analysis indicates that due to cloud data center homogeneity, a suitable solution can be found without analyzing all servers, switches, and paths. Due to the larger set of constraints introduced by an SDN-based model, QVIA-SDN has the longer runtime. However, given the volume of constraints and the complexity of the process, the runtime of 30 seconds in the worst case can be considered acceptable for some cloud providers.

The number of constraints on LP, even after relaxing some binary and integer variables, represents a dimensionality barrier justifying the increasing on the allocation time. However, the analysis with different pruning configurations (60%, 80%, and 100% on Fig. 10) indicates that for both Fat-Tree scenarios ($k = 4$ and $k = 8$), QVIA-SDNcan increase the number of candidates under analysis to find solutions with lower internal latency (Fig. 6), demonstrating the efficiency of all remaining algorithms composing QVIA-SDN. Although using the same network topology from simulation with uniformly distributed loads on EC2-based scenario, the configuration of virtual machines is different. Instead of selecting values from a predefined interval, the EC2-based scenario investigates the allocation using fixed virtual machine flavors. Consequently, the LP solver can identify identical values to prune the search space on-the-fly, despite the QVIA-SDNconfiguration. Finally, it is worthwhile to mention that the VI allocation on SDN-based data centers is a NP-Hard problem. The addition of constraints to achieve a latency-efficient allocation, specifically the network constraints, is a promising approach to providers and tenants. However, this proposal open opportunities for further research on search space optimizations. A possible research line can investigate the use of grouping techniques to simplify the network topology.

Physical candidates pruning is a promising approach. The pruning technique applied to decrease the number of physical candidates for hosting VIs is a promising approach. For small-scale data centers ($k = 4$, Figures 5(a), 12(a) and 12(b)), the key rejection reason is the servers, switches and network load. When the number of candidates is decreased, some requests are rejected as the LP solver is unable to find a solution, even when variables are relaxed. For datacenters with $k = 8$ configuration (Figures 5(b), 12(c) and 12(d)), the rejection is mainly caused by switches load even when all resources are considered as possible candidates.

7 Considerations

SDN has being applied by public and private cloud providers for internal IaaS management and service provisioning. Despite all benefits introduced by SDN decoupled management of data and control planes, the paradigm brought a set of challenges related to virtual infrastructures provisioning. Specifically considering the allocation of physical resources for hosting VIs, SDN introduced new constraints on physical switches and network topology, such as size-limited flow-tables, increased latency to the controller when a flow-table miss is triggered, and long hosting paths.

In this context, the present work formulates the online VI allocation on SDN-based cloud data centers as an optimal MIP considering all traditional aspects and SDN challenges. Additionally, a mechanism was proposed to relax the MIP constraints obtaining a linear program as well as a rounding heuristic. The proposed mechanism, QVIA-SDN, was compared with two baseline approaches, and the results highlight that for allocating QoS-aware VIs atop SDN-based cloud data centers, the traditional allocation mechanisms, without considering latency requirements and SDN configuration, result in VIs provisioned with higher internal latency. Moreover, in cloud data centers with homogeneous resources, the number of physical candidates accounted for finding a solution can be pruned without compromising the provider and tenants perspectives. The promising results obtained by modeling SDN resources and constraints indicate some future directions. Initially, the logically centralized knowledge of an SDN controller can be used to share residual bandwidth among cloud tenants [30], while a second line indicates a formulation considering availability and reliability requirements.

Acknowledgements This research was supported by the UDESC PROMOP program and was developed at the LabP2D.

References

1. Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, August 2008.
2. Ali Al-Shabibi, Marc De Leenheer, Matteo Gerola, Ayaka Koshibe, Guru Parulkar, Elio Salvadori, and Bill Snow. Openvirtex: Make your virtual sdns programmable. In *Proceedings of the 3rd Workshop on Hot Topics in Software Defined Networking*, HotSDN '14, pages 25–30. ACM, 2014.
3. Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. Towards predictable datacenter networks. *SIGCOMM Comput. Commun. Rev.*, 41(4):242–253, August 2011.
4. Der-San Chen, Robert G Batson, and Yu Dang. *Applied integer programming: modeling and solution*. John Wiley & Sons, 2010.
5. M. Chowdhury, M.R. Rahman, and R. Boutaba. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on Networking*, 20(1):206–219, Feb 2012.
6. N. M. Mosharaf Kabir Chowdhury and Raouf Boutaba. Network virtualization: State of the art and research challenges. *Comm. Mag.*, 47(7):20–26, July 2009.
7. NM Mosharaf Kabir Chowdhury and Raouf Boutaba. Network Virtualization: State of the Art and Research Challenges. *Communications Magazine, IEEE*, 47(7):20–26, 2009.
8. Bryce Cronkite-Ratcliff, Aran Bergman, Shay Vargaftik, Madhusudhan Ravi, Nick McKeown, Ittai Abraham, and Isaac Keslassy. Virtualized congestion control. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 230–243, New York, NY, USA, 2016. ACM.
9. G.A. De S Cavalcanti, R.R. Obelheiro, and G. Koslovski. Optimal resource allocation for survivable virtual infrastructures. In *Design of Reliable Communication Networks (DRCN), 2014 10th International Conference on the*, pages 1–8, April 2014.
10. Felipe Rodrigo de Souza, Charles Christian Miers, Adriano Fiorese, and Guilherme Piegas Koslovski. QoS-Aware Virtual Infrastructures Allocation on SDN-based Clouds. In *17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2017.
11. Mehmet Demirci and Mostafa Ammar. Design and analysis of techniques for mapping virtual networks to software-defined network substrates. *Computer Communications*, 45:1 – 10, 2014.

12. Dmitry Drutskoy, Eric Keller, and Jennifer Rexford. Scalable network virtualization in software-defined networks. *Internet Computing, IEEE*, 17(2):20–27, March 2013.
13. Sevil Drxler, Holger Karl, and Zoltn dm Mann. Joint Optimization of Scaling and Placement of Virtual Network Services. In *17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, May 2017.
14. A. Fischer, J.F. Botero, M. Till Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *Communications Surveys Tutorials, IEEE*, 15(4):1888–1906, Fourth 2013.
15. Chuanxiong Guo, Guohan Lu, Helen J. Wang, Shuang Yang, Chao Kong, Peng Sun, Wenfei Wu, and Yongguang Zhang. 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, 2010. ACM.
16. Keqiang He, Eric Rozner, Kanak Agarwal, Yu (Jason) Gu, Wes Felter, John Carter, and Aditya Akella. Ac/dc tcp: Virtual congestion control enforcement for datacenter networks. In *Proceedings of ACM SIGCOMM 2016 Conference*, pages 244–257, New York, NY, USA, 2016. ACM.
17. Brendan Jennings and Rolf Stadler. Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management*, pages 1–53, 2014.
18. Nanxi Kang, Zhenming Liu, Jennifer Rexford, and David Walker. Optimizing the "one big switch" abstraction in software-defined networks. In *Proceedings of the 9th ACM CoNEXT '13*, pages 13–24, New York, NY, USA, 2013. ACM.
19. G. Koslovski, S. Soudan, P. Gonalves, and P. Vicat-Blanc. Locating virtual infrastructures: Users and inp perspectives. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 153–160, May 2011.
20. D. Kreutz, F. M. V. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig. Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):14–76, January 2015.
21. Bo Li, Songtao Guo, Yan Wu, and Defang Liu. Construction and resource allocation of cost-efficient clustered virtual network in software defined networks. *Journal of Grid Computing*, 15(4):457–473, Dec 2017.
22. Sunilkumar S. Manvi and Gopal Krishna Shyam. Resource management for infrastructure as a service (iaas) in cloud computing: A survey. *Journal of Network and Computer Applications*, 41:424 – 440, 2014.
23. R. Mijumbi, J. Serrat, J. Rubio-Loyola, N. Bouten, F. De Turck, and S. Latre. Dynamic resource management in sdn-based virtualized networks. In *Proceedings of 10th CNSM*, pages 412–417, Nov 2014.
24. Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, and Raouf Boutaba. A path generation approach to embedding of virtual networks. *CoRR*, abs/1509.07684, 2015.
25. Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. Portland: A scalable fault-tolerant layer 2 data center network fabric. *SIGCOMM Comput. Commun. Rev.*, 39(4):39–50, August 2009.
26. Ramon de Oliveira and Guilherme Koslovski. A tree-based algorithm for virtual infrastructure allocation with joint virtual machine and network requirements. *International Journal of Network Management (IJNM)*, 2016.
27. Uroš Paščinski, Jernej Trnkoczy, Vlado Stankovski, Matej Cigale, and Sandi Gec. Qos-aware orchestration of network intensive software utilities within software defined data centres. *Journal of Grid Computing*, 16(1):85–112, Mar 2018.
28. Valerio Persico, Pietro Marchetta, Alessio Botta, and Antonio Pescape. Measuring network throughput in the cloud: The case of amazon {EC2}. *Computer Networks*, 93, Part 3:408 – 422, 2015. Cloud Networking and Communications {II}.
29. Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, Keith Amidon, and Martin Casado. The design and implementation of open vswitch. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 117–130, Oakland, CA, 2015. USENIX Association.
30. Lucian Popa, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. Faircloud: Sharing the network in cloud computing. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, pages 22:1–22:6, New York, NY, USA, 2011. ACM.
31. Matthias Rost, Carlo Fuerst, and Stefan Schmid. Beyond the stars: Revisiting virtual cluster embeddings. In *In Proc. ACM SIGCOMM Computer Communication Review (CCR*, 2015.

32. Rob Sherwood, Glen Gibb, Kok kiong Yap, Martin Casado, Nick Mckeown, and Guru Parulkar. Flowvisor: A network virtualization layer. Technical report, 2009.
33. Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru Parulkar. Can the production network be the testbed? In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, pages 1–6. USENIX, 2010.
34. William Stallings. *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud.* Addison-Wesley Professional, 1st edition, 2015.
35. Feng Tao, Bi Jun, and Wang Ke. Allocation and scheduling of network resource for multiple control applications in sdn. *Communications, China*, 12(6):85–95, June 2015.
36. Tram Truong Huu, Guilherme Koslovski, Fabienne Anhalt, Johan Montagnat, and Pascale Vicat-Blanc Primet. Joint elastic cloud and virtual network framework for application performance-cost optimization. *Journal of Grid Computing*, 9(1):27–47, 2011.
37. Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: substrate support for path splitting and migration. *SIGCOMM Comput. Commun. Rev.*, 38(2):17–29, 2008.