QoS-Aware Virtual Infrastructures Allocation on SDN-based Clouds

Felipe Rodrigo de Souza, Charles Christian Miers, Adriano Fiorese, Guilherme Piegas Koslovski
Graduate Program in Applied Computing – Santa Catarina State University – Joinville – Brazil
dcc6frs@joinville.udesc.br, {charles.miers, adriano.fiorese, guilherme.koslovski}@udesc.br

Abstract—**Virtualization of computing and communication infrastructures were disseminated as possible solutions for networks evolution and deployment of new services on cloud data centers. Although promising, their effective application faces obstacles, mainly caused by rigidity on the management of communication resources. Currently, the Software-Defined Networks (SDN) paradigm has been popularizing customization and flexibility in network management due to separation of control and data planes. However, benefits introduced by SDN are not trivially applied to Virtual Infrastructures (VIs) provisioning on SDN-based cloud providers. An allocation mechanism needs joint information of control and data planes in order to deliver Quality-of-Service (QoS)-aware mappings while achieving provider objectives. In this work, we formulate the online VI allocation on SDN-based cloud data centers as a Mixed Integer Program (MIP). Following, integer constraints are relaxed to obtain a linear program, and rounding techniques are applied. The mechanism performs VI allocation considering latency, bandwidth, and virtual machine requirements. The results indicate that the VIs mean internal latency can be reduced while simultaneously enforcing other QoS constraints.**

1. Introduction

Cloud Computing has revolutionized the provisioning of computing and networking services. Particularly, Infrastructure-as-a-Service (IaaS) providers have used virtual networking technologies to deliver VIs [1]. A VI is composed of Virtual Machines (VMs) interconnected by virtual networking resources, where the number of virtual resources (*e.g.,* machines, switches, and links) and their configuration (*e.g*, processing and bandwidth) can be dynamically adjusted based on hosted application requirements.

Network configuration and management are critical tasks in IaaS clouds. VI-hosted applications are responsible for large volumes of network traffic while an expressive portion of its running time is due to network activity. For instance, a Facebook cluster can go up to 33% of its running time only making data transfers [2] while under-provisioned virtual networks can drastically affect hosted application performance [3]. The application of network virtualization techniques is a driven force to efficiently provide VIs [4] [5]. Recently, existing VIs deployment obstacles caused by the inflexibility of network equipments have been addressed by the SDN concept. SDN promoted network reorganization removing the exclusive control traditionally built-in on network equipment. A logically-centralized controller, with knowledge of the data center topology and load, is responsible for network-traffic engineering [6]. Although promising, the application of SDN techniques as a driven force to implement virtualized cloud data centers is a challenging task. SDN has added new dimensions on data center management: flows forwarding delay; dynamic virtual topology creation; bandwidth sharing; CPU isolation on switches; and switching tables sharing [7] [8]. Moreover, tenants can optionally deploy private SDN controllers to orchestrate VIs.

In this context, the present work addresses the allocation of SDN-based resources for hosting VIs. The VIs allocation problem, with constraints on virtual resources and topology, belongs to the set of NP-hard problems as other problems already proven to be in this set may be reduced to it [9] [10]. Specifically, an SDN-based formulation increases the VI allocation complexity in three main axis. *(i)* New constraints on physical switches capacities are introduced: SDN switches have a size-limited flow-table (eventually, allocated flows must be replaced by new flows). *(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 Round-Trip Time (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.

Literature comprises several mechanisms to find optimal and approximated solutions for VI allocation [10] [11] [12]. In most cases, mechanisms solve only the allocation of virtual networks without tackling virtual processing and switching resources. Some proposals have advanced the field by adapting virtual network formulations to SDN-based data centers [13] [14] [15] [16] [17], despite not considering the particularities of IaaS cloud data centers. In short, we make three main contributions in this paper:

- We formulate the online VI allocation in SDN-based cloud data centers as an optimal MIP. Our formulation considers the main management challenges introduced by SDN, modelling 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.

- Results simulating the application of the proposed mechanism for allocating VI requests atop a cloud data center interconnected by a fat-tree topology [18]. The evaluation quantifies provider-based metrics and tenants perspective.

The remainder of this paper is organized as follows. Section 2 outlines challenges on VI allocation atop SDN-based data centers. Section 3 discusses related work. Section 4 defines and formulates the VIs allocation problem. Section 5 details the proposed MIPs formulation while Section 6 discusses the proposed heuristic based on relaxing constraints. Experimental results are presented in Section 7, while final considerations and future works are listed in Section 8.

2. SDN-based VI allocation

A cloud management framework relies on allocation algorithms to efficiently identify physical resources for hosting VIs. Usually, there are constraints that need to be satisfied during the allocation process in order to guarantee the physical infrastructure capability to provide the requested virtual resources [10]. Indeed, the problem of cloud data center resources allocation to host VIs is a hard one due to its computational expensiveness and complexity, and the need of taking into account a wide range of constraints originated from different tenants and providers. On IaaS providers, the number of servers employed for composing a data center appears as a challenging aspect [19], even performing a prune on physical candidates by restricting the search to a given data center, region, or zone. Moreover, VI allocation problem is exacerbated by the multi-criteria constraints that must be satisfied. A VM may require a specific 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 can be seen as an extension of IaaS paradigm (including network resources) the allocation problem can be examined as a virtual network embedding formulation with additional nodes constraints [20] [10].

The VI allocation is abstracted as a graph embedding problem: vertices represent compute and network equipments, while edges denote links and paths. Each graph element has a set of requirements or capacities associated with. Virtual graphs carrying out tenant requirements must be placed atop a physical graph, which is representing the cloud data center infrastructure.

SDN is a promising technology for isolating VIs provisioning. Indeed, a VI can be provisioned with a private controller (or an interface for communication with physical controllers) used for internal management (*e.g.*, load balancing, virtual topology segmentation) [7]. Considering QoS requirements, SDN introduces opportunities for latency and bandwidth control by decoupling data and control planes. Although innovative, this approach leads to latency increase on virtual resources communication as the traversed path of a flow is extended. When a flow is proactively configured along the path, latency is increased by a single RTT to controller. The worst case can be observed on frequent changing scenarios: as flows are periodically removed from switches' flow-tables, controllers interference is eventually required [7]. As this fact imposes a challenging on allocation of latency-sensitive applications, we claim that an allocation model must consider switches' flows-table as a shared resource for allocating QoS-aware VIs.

3. Related work

The literature defines that virtual network provisioning on IaaS cloud data centers need to be driven by VM importance (*e.g.,* instance type) or specifically defined (virtual links). Based on this train of thought, policies can be locally (per physical links, or congested paths) or globally applied (considering the data center topology) [21], [22]. Popa *et al.* [23] proposed a set of allocation policy that aims to approximate network sharing proportionality and can only be achieved in a controlled scenario, such as SDN, as it requires coordination between network controller and virtual resources. Policies are controlled by a parameter indicating that a communication has the same configuration and importance through the data center network paths. This parameter is used to compute sharing on physical paths which are hosting virtual links and it is aligned with our proposal: we use a virtual link configuration between two VMs (bandwidth and latency) as a specification of requirements.

Specifically considering the mechanisms for selecting physical resources to host VMs and links, literature exposes proposals on optimal formulations and approximation heuristics [10]. In general, each proposal lays down a specific and restricted usage scenario, usually focusing on data center metrics optimization, such as fragmentation, cost, revenue, acceptance ratio, among others. Chowdhury *et al.* [11] proposed the joint allocation of virtual processing and communicating resources, as discussed here.

The SDN-aware VI allocation is an open challenge for IaaS cloud providers. [7] identified the key challenges related with SDN-based virtual resource provisioning, pointing out the importance of flow tables and controllers sharing among hosted VIs. Moreover, they proposed a hypervisor for instantiating virtual networks, with similar management approaches to those implemented by VMs hypervisors. In addition, [24] proposed a framework for SDN-assisted network virtualization. We propose a QoS-aware allocation algorithm taking into account the particularities and challenges that the authors [24] have identified while filling out a provisioning gap identified in both works [7] [24].

SDN controller placement was studied in [17]. Authors identified that latency to controller is a key factor for final applications performance. Complementary, [15] focused on virtual controllers provisioning, highlighting that hosted virtual networks can have distinct addressing schemes and routing policies. Some mechanisms for allocating network bandwidth considering SDN resources were proposed in [14] [13]. Features individually identified by both works should be combined when allocating resources to host VIs. In this work, SDN-aware constraints are proposed in a MIP formulation to achieve bandwidth and latency control. Also, the optimal MIP proposed in Section 5 considers that virtual switches can be reserved and individually managed by cloud tenants. In this context, we concentrated on QoS-aware VI allocation atop SDN-based cloud data centers, considering the challenges and research opportunities indicated by the literature. Moreover, we formulate the allocation problem as a joint allocation of VMs, switches, and links, claiming that performance of cloud-hosted applications can be impacted by networking allocation policies.

4. Problem formulation

4.1. Cloud data center and VI requests

Both cloud data center and VI requests are represented by weighted undirected graphs. 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 1 resumes the notation used along this paper.

Notation	Description
$G^s(N^s_h, N^s_n, C^s, E^s)$	cloud data center graph
N^s_h	physical servers
N^s_n	physical SDN-based 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-based 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 $\Omega(j)$

TABLE 1. NOTATION USED ALONG THIS PAPER. i AND j REPRESENT VIRTUAL RESOURCES WHILE u AND v ARE USED FOR PHYSICAL ONES.

As networking is a key aspect for hosted applications performance, cloud providers are moving towards VMs provisioning with QoS network requirements. In this context, VI requests can be designed based on IaaS and Network-as-a-Service (NaaS) scenarios as depicted on Figure 1.

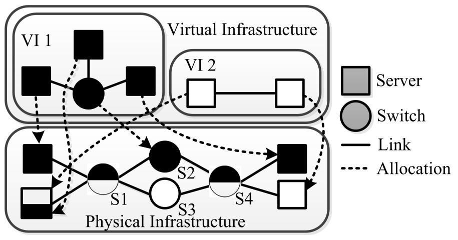


Figure 1. *E.g.,* VI requests allocation atop an SDN-based cloud provider.

- *IaaS-only requests* represent the traditional cloud requests for VMs without specifying the intermediate hops and configuration of virtual networks. However, we argue that in short time a tenant may be able to request a maximum latency between two VMs as well as a minimum end-to-end bandwidth for another pair, instead of just specifying the data center, region or zone locations. This request is exemplified by the *VI 2* (Fig. 1), in which two VMs must be provisioned with network QoS. It is worthwhile to note that the networking configuration used to provide the connectivity is a private cloud provider information. Indeed, the physical path (including switches) enabling VMs communication is abstracted from tenants.
- *Combined IaaS / NaaS requests* enable a full description of VMs and interconnecting virtual network topology. In this request, a tenant can detail all intermediate virtual switches and the required QoS configuration among them. As exemplified by *VI 1* request (Fig. 1), a cloud provider can use more physical resources for hosting a virtual one: the VI requested just one switch, while the cloud provider employed three switches for composing the VI. In this scenario, only the required switch is exposed to tenant with SDN management capabilities [7].

A cloud provider can explore SDN technology to enforce the QoS requirements for both request sets.

4.2. Allocating physical resources to host VIs

VI requests are individually processed by a cloud orchestration framework which has to determine whether to accept or not the request. Although any VI component may change during its lifespan using the elastic provisioning of virtual resources, its reconfiguration is not discussed in the present work as it is not directly represented in an online formulation. We argue that initial provisioning and elastic reconfiguration must be independently analyzed and deployed. The allocated resources are released once the tenant terminates the VI.

The allocation of VI requests onto a cloud data center can be decomposed into nodes and links assignments [25] [11]. A map of VMs onto physical servers is given by $M_h : N^v_h \mapsto N^s_h$, while for virtual switches on SDN-based equipments is denoted as $M_n : N^v_n \mapsto N^s_n$, with $M_h(i) \in N^s_h$ and $M_n(j) \in N^s_n$. 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 $M_e(ij) \subseteq P^s(M_h(i) \cup M_n(i), M_h(j) \cup M_n(j))$. For hosting a virtual resource, the physical node must have a residual capacity greater or equal than the virtual requested. Residual capacity represents the remaining available resources on physical equipments (*e.g.,* servers, switches, links, and paths) not compromised on hosting virtual entities. For VMs and switches, $c_i \leq R(M_h(i))$ and $c_j \leq R(M_n(j))$, respectively. Similarly, for virtual links $c_{ij} \leq R(M_e(ij))$.

4.3. Cloud provider objectives

Our main interest in this paper is to allocate VI delivering the QoS requirements. In addition, we want to decrease the allocation cost while simultaneously increase the provider revenue. We define the cost for hosting a VI proportionally to physical capacity reserved to Service Level Agreement (SLA) assurance as given by Eq. (1), in which $|ij|$ represents the length of path hosting ij in terms of hops [12] [11]. Complementary, the revenue for hosting a VI is given by Eq. (2).

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

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

VI provisioning quality is quantified by the mean latency between communicating virtual resources, as denoted by Eq. (3), 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.

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

5. Optimal MIP for QoS-aware VI allocation

5.1. Selecting candidates to host virtual resources

On public IaaS providers, tenants select the region/zones for VMs instantiation[1]. It is plausible to state that this type of option is an approximation for VM placement based on geographical location [26]. Physical candidates for hosting a virtual resource i are represented by the set $\Omega(i)$. Each virtual resource has a geographical location requirement (zone or region) denoted by $loc(i)$. Any physical resource placed on location $loc(i)$ is a candidate for hosting i.

We claim that besides zones and regions, users must be able to specify real network requirements (bandwidth and latency). For communication-intensive applications, latency is a critical factor [27]. Motivated by the use of network virtualization techniques on IaaS data centers, latency and network performance are not completely dependent on physical location [19]. In this regard, instead of performing the candidates selection only based on a fixed location as traditionally used on allocation mechanisms [28] [11], we introduce a selection based on end-to-end latency.

1. 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.

Candidates for hosting virtual switches with latency requirements. When the VI request includes latency requirements for links from/to virtual switches, the subset of physical candidates for hosting a switch i is composed of equipments having output links capable of hosting the worst case latency requirement of i. In other words $\Omega(i) = \{u \in N_n^s | max(lat(u,v)) < max(d_{ij})\}; \forall v \in adj(u); \forall j \in adj(i)$, 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. In this case, 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: for this subset, 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, $\Omega(i) = \{u \in N_h^s | lat(u,v) \leq d_{ij}\} \forall v \in N_h^s$.

Following [11], in order to perform a joint allocation of edges and vertices, each virtual resource i is assigned to the physical graph, and interconnected with their candidates through a temporary link with infinity capacity and no communication latency. The resulting augmented graph is denoted by $G^{s'}(N^{s'}, C^s, E^{s'})$. In this sense, $N^{s'} = N_h^s \cup N_h^v \cup N_n^s \cup N_n^v$; and $E^{s'} = E^s \cup \{iu \,|\, i \in N_h^v, u \in \Omega(i)\} \cup \{ju \,|\, j \in N_n^v, u \in \Omega(j)\}$. SDN controllers ($C^s$) are not affected by the augmented graph. Fig. 2 exemplifies an augmented graph by connecting *VI 1* request (Fig. 1) to physical resources by means of temporary links.

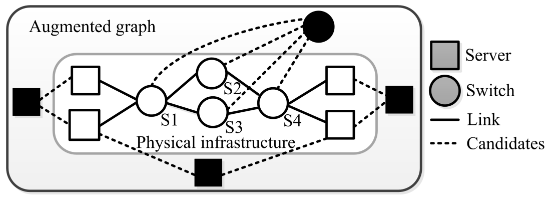


Figure 2. An augmented graph combining virtual and physical resources.

5.2. Variables and objective

A combination of three variables is used to represent a possible map solution for virtual nodes, switches, and links with QoS requirements. The variable f_{ijuv} accounts the amount of ij flows allocated on physical link uv, while the variable x_{uv} belongs to a binary domain to indicate the presence of a flow (a virtual link request) from u to v. It is set to 1 if $\sum_{ij \in E^v}(f_{ijuv} + f_{ijvu}) > 0$, otherwise 0. Representing switch flow-tables and the controllers usage, e_{ijpu} is a binary variable to indicate the presence of ij flow on SDN controller (set to 1), or pointing that flow-entry to the path p is hosted on switch u (set to 0).

In order to achieve the main objective (Section 4.3) we use a modified version of Equations (1), (2), and (3) to compose the objective function (Eq. (4)). By minimizing the objective function, we envisage to decrease the substrate cost for hosting VIs, mainly for virtual links allocation and flow-tables reservation. As previously proposed by literature, by dividing the cost with the residual capacity of nodes, switches and links, physical resources with more residual capacity are preferred facing less residual capacity ones [11]. Parameters α_{uv}, β_u, and γ_e control the importance of a resource representing the cloud provider view about data center usage, ranging between 1 and $R(.)$, while δ is a positive number (avoid division by zero).

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

5.3. Constraints

For guaranteeing the QoS established on SLA, a set of capacity, flow-related, meta and binary constraints must be satisfied by the allocation mechanism.

Servers and links capacity constraints. Eq. (5) defines capacity constraints for physical links while Eq. (6) applies to physical servers. In short, physical resources must have enough capacity for hosting the requests.

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

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

SDN-related constraints. SDN-based switches have a specificity on forwarding table capacity: even when a flow is passing through the switch (identified by variable x), the corresponding flow-table entry maybe only present on SDN controllers (identified by variable e). In this sense, Eq. (7) indicates that the residual capacity of a switch u must be larger enough to host all flow-table entries, however, those entries hosted by a controller are not considered. s_{ij} and t_{ij} represent the source and target nodes of a virtual link ij.

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

Flow-related constraints. Each virtual link request represents a flow that must be allocated and forwarded atop a physical infrastructure. Eq. (8) guarantees that for each virtual link ij, all flows starting from s_{ij} must be equals as the requested virtual link capacity, while Eq. (9) has the same meaning for a flow arriving at t_{ij}. Complementary, Eq. (10) is responsible for the flow forwarding on an SDN topology. Each intermediate physical resource must forward exactly all flows received to the target resource.

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

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

$$\sum_{v \in N^{s'}} f_{ijuv} - \sum_{v \in N^{s'}} f_{ijvu} = 0 \qquad \forall ij \in E^v, \forall u \in N^{s'} \setminus \{s_{ij}, t_{ij}\} \quad (10)$$

Latency constraints. A tenant may compose an SLA informing latency requirements. Thus, for delivering the QoS, a provider must host virtual links atop physical paths respecting the requested configuration (d_{ij}, for a virtual link ij). Eq. (11) and (12) guarantee that the latency of a physical path hosting a virtual link is lower than the requested value even when the flow-table entry is placed at the controller.

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

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

Meta and binary constraints. Eq. (13) ensures that a virtual resource is allocated by a single physical resource, while Eq. (14) ensures that x is set when there is any flow passing through. Eq. (15) to (17) indicate the variable domains.

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

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

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

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

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

6. QVIA-SDN mechanism

Solving a MIP is known to be computationally intractable [29] [11]. Consequently the proposed optimal MIP for QoS-aware VI allocation is practically infeasible. Coping with, a set of techniques are applied for relaxing the integer constraints obtaining an Linear Program (LP), and to prune the number of physical candidates and paths, decreasing the search space. After solving the LP, the approximated solution is treated by a rounding heuristic. The LP, pruning and rounding techniques compose QVIA-SDN (QoS-Aware VI Allocation on SDN-based data centers).

6.1. Relaxing variables

For obtaining an LP, constraints (16) and (17) are relaxed originating Eq. (18) and (19), respectively, with domains between 0 and 1. The objective function and all other constraints remain unchanged.

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

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

6.2. Pruning physical candidates

Cloud data centers are usually composed of a set of homogeneous resources interconnected by structured network topologies [27]. Independently of the topology, physical servers may be grouped by some similar aspects (*e.g.,* bandwidth, latency and processing power). This information can be used to compose groups of candidates [2]. Following this line, QVIA-SDN decreases the number of physical candidates by enforcing a maximum percentage for each data center region. This information can be adjusted by providers according to the number of available resources per region and zone. One may expect that by limiting the number of candidates, the efficiency of the algorithm should be impacted by the constrained solution space. The impact of this parameterization is discussed in Section 7.

6.3. Decreasing the number of physical paths

Accounting and controlling all paths between physical resources composing a data center is computationally expensive and practically infeasible for online VI allocation. As the resulting LP requires the set P^s to find an approximated solution, QVIA-SDN employs a networking path pruning for decreasing the number of candidates. Data centers topology are commonly composed of multiple paths between servers for load balancing, fault-tolerance, and reliability purposes [27], [18]. QVIA-SDN explores this fact accounting just a subset of paths between all physical resources for composing P^s. This temporary pruning only hides the existence of some paths between physical resources. Any reliability algorithm or backup-traffic engineering can be latter applied. In other words, P^s is initially composed of *(i)* one shortest path and *(ii)* one small latency path between all pairs of physical candidates (servers and switches), in which *(i)* is different from *(ii)*.

6.4. Rounding heuristic

When performing the LP relaxation process, the values obtained for variables x and e are no longer binary and the correlation between f, x and e is lost. Therefore, QVIA-SDN employs a heuristic to evaluate the obtained values and to approximate a possible solution. The heuristic is composed of two steps, as given by Algorithms 1 and 2.

The first step (Alg. 1) is based on D-ViNE [11]: initially, an augmented graph connecting the virtual resources to their candidates is created, following the definitions from Section 5.1, and then the LP with relaxed variables is solved (lines 1 and 2). The next step identifies a suitable hosting candidate for each virtual resource. In this sense, the p_z

```
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 $k \in N^v_h \cup N^v_n$ 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 $k \in N^v_h$ 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 [11].

is calculated for all physical candidates to host a virtual resource k (lines 3 to 16), defined as a weighted product of x_{kz} and the total flow passing through kz in both directions. In other words, $p_z = \alpha(\sum_{ij \in E^v} f_{ijkz} + f_{ijzk}) + (1-\alpha)x_{kz}$. The rationale behind this approach is to reconstruct the correlation between f and x jointly identifying the mapping of vertices and edges. The weight α is used to guide the heuristic preference (network or servers) when combining both information. The candidate with the highest p_z is selected to allocate the virtual element. When no candidates are identified, VI allocation is rejected (lines 8 to 9).

After identifying a suitable mapping for virtual machines and switches, the appropriate paths for hosting all virtual links are accounted by QVIA-SDN considering the SDN particularities (controllers, switches and flow tables). The Deterministic Path Search (DPS) is described in Algorithm 2. DPS is based on e values, latency and bandwidth requirements for identifying whether a physical path can host a virtual link ij, aiming to decrease the use of switches by mapping flow entries at controller, when possible.

In this sense, when two communicating virtual resources are mapped on a single physical host, it is assumed that the host has enough capacity for providing bandwidth and latency requirements (lines 2 to 4). A discussion on internal virtualization hypervisor allocation is let as future work. All possible hosting paths for a virtual link ij (line 5) are analyzed considering the presence or absence of correspondent flow-table entries (lines 8 to 14). If there is any value in e, QVIA-SDN accounts the latency to the controller instead of allocating a flow-table entry at the corresponding switch.

Physical paths with latency higher than the requested configuration d_{ij}, even with flow-table entries allocated in all switches along the path, are discarded. A similar rationale is applied to bandwidth requirements ignoring paths without the minimum requirements. However, in some cases a path can be reconfigured to accommodate the requested latency

```
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.

configuration. In this case, QVIA-SDN verifies whether there is one combination that can allocate the virtual link with the minimum number of physical switches and hops and still respect the latency request by solving a knapsack problem involving the resources along the path. Among all paths, we select those with minimum number of hops (length of a path) and maximum number of flow-table entries hosted by the controller (decreasing the switches load). It is worthwhile to mention that such process still respect the bandwidth requirements. Finally, in the absence of a physical path for hosting a virtual link, VI is rejected.

7. Evaluation

The evaluation quantifies the data center usage (provider's perspective) as well as the QoS delivered to tenants. Initially, the metrics and the simulation scenarios are detailed, and latter the results comparing QVIA-SDN with two baseline mechanisms are discussed.

7.1. Metrics

For representing the cloud provider objectives (Section 4.3), five metrics were selected. *(i) Revenue-to-cost ratio* gives an insight of how much a provider will gain by accepting a VI request. *(ii)* Cloud data center *fragmentation* indicates the percentage of active resources that are hosting VIs, figured out by dividing the number of active resources by the total number of available resources. *(iii)* The *mean runtime to allocate* a VI request. *(iv) Acceptance ratio* quantifies the percentage of successfully allocated VIs. *(v)* The *mean latency* of a provisioned VI, quantified by summing up the latency of physical paths hosting virtual links and latter dividing by the number of requested virtual links.

7.2. Simulation scenarios

QVIA-SDN and a discrete event simulator were implemented in Java v1.8, using CPLEX optimizer (v12.6.1.0) for solving the LP[2]. Experiments were performed on Intel Xeon E5-2620 2.0GHz - 24 cores, 196GB (DDR3) RAM. For analyzing QVIA-SDN applicability on cloud scenarios, the fat-tree topology was selected to represent the cloud data center [18], while two VI topologies commonly used in public and private providers were composed: multi-tiered and virtual private clouds.

Fat-tree topology. A fat-tree is composed of k pods, each containing two layers of $k/2$ switches. In short, a fat-tree build with k-port switches supports up to $k^3/4$ servers. In this paper, two configurations are considered, with $k = 4$ and $k = 8$. For representing the physical capacity of servers, switches and links, absolute values were defined: 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 SDN controller is 2 ms. Core switches are directly connected with the SDN controller, while other switches are connected by logical paths. The hierarchical organization of regions and zones into a fat-tree is defined as follow: each zone is represented by a pod, while a pair of zones composes a region. The predefined values are useful for simulation purposes and can be replaced by real values according to the cloud data center.

Multitiered VI requests. A large fraction of cloud tenants organizes the VMs following a n-layers topology [30], comprising a load balancer in charge of distributing requests for a set of web servers, that eventually query a database. For the simulation purpose, each layer was defined with a dedicated virtual switch for requesting network QoS parameters. The number of VMs for web servers and databases is uniformly distributed between 10 and 20, while communication between virtual resources is represented by virtual link requests.

Virtual Private Clouds (VPC). Amazon EC2 introduced the dynamic provision of Virtual Private Clouds (VPCs)[3] composed of a subset of access point rules and a set of VMs attached to it, composing a private Local Area Network (LAN) that is managed by the tenant. For composing VPC requests, a set of VMs is connected to an SDN-based virtual switch. The number of VMs follows an uniform distribution between 5 and 10 elements.

For 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 an uniform distribution). As cloud data centers are organized in regions and zones, the geographical location of VIs is defined by an initial random selection of region,

2. IBM CPLEX Optimizer: https://www.ibm.com/software/commerce/optimization/cplex-optimizer/.

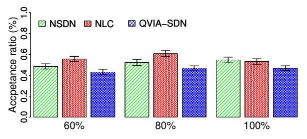
3. Virtual Private Cloud (VPC): https://aws.amazon.com/vpc/.

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 multi-tiered) 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).

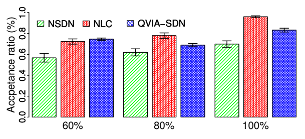
7.3. Simulation results

The results show sample means with 95% confidence intervals. QVIA-SDN is compared with two baseline algorithms. First, a formulation without SDN knowledge and latency control, labeled as NSDN, represents a common approach on literature [11]. In order to isolate the QVIA-SDN latency-control overhead, a version without latency optimization constraints is presented and identified by 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 6.2. Based on empirical observations, QVIA-SDN parametrization is $\alpha = 0.9$ (Section 6.4) and $\beta_u = \gamma_e = \alpha_{uv} = 1$ (Section 5.2).

7.3.1. Acceptance ratio. The acceptance ratio for $k = 4$ and $k = 8$ scenarios is presented in Figures 3(a) and 3(b), respectively. Due to limited number of available physical resources on $k = 4$ configuration, Fig. 3(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. 3(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.



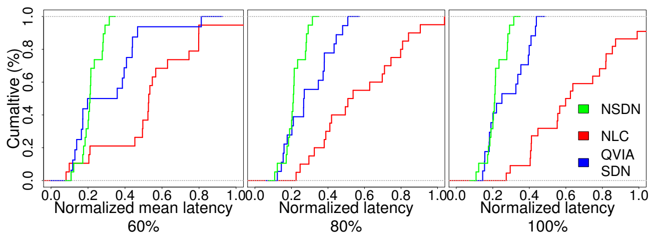
(a) $k = 4$.



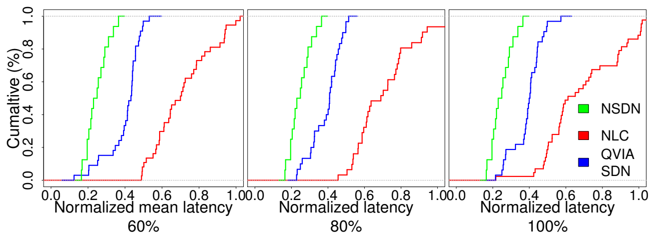
(b) $k = 8$.

Figure 3. VI acceptance ratio.

7.3.2. Average latency. Fig. 4 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. 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, Fig. 5 indicates that QVIA-SDN presents small variability on latency compared to NLC.

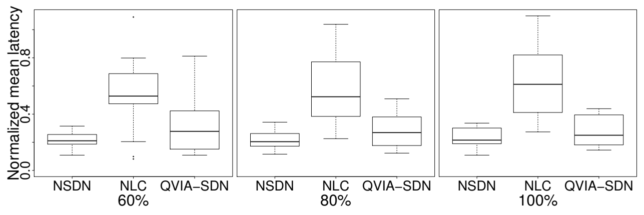


(a) $k = 4$.

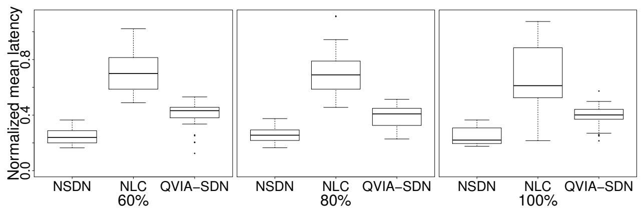


(b) $k = 8$.

Figure 4. Cumulative distribution of normalized mean latency.



(a) $k = 4$.

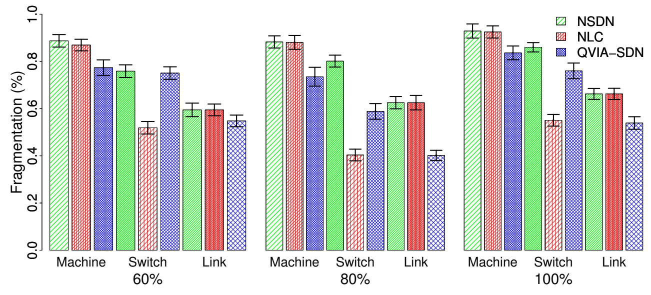


(b) $k = 8$.

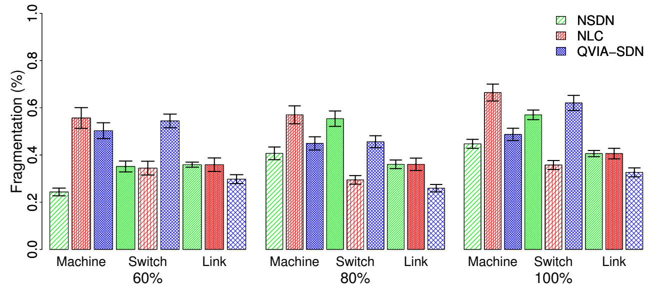
Figure 5. Mean latency variability.

7.3.3. Fragmentation. Figures 6(a) and 6(b) present the fragmentation for $k = 4$ and $k = 8$, respectively. QVIA-SDN tends to condense virtual resources atop the data center. For 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.

7.3.4. Revenue-to-cost ratio. Figures 7(a) and 7(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.

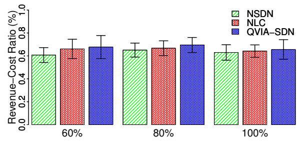


(a) $k = 4$.

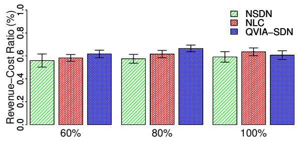


(b) $k = 8$.

Figure 6. Data center fragmentation.



(a) $k = 4$.



(b) $k = 8$.

Figure 7. Revenue-to-cost ratio.

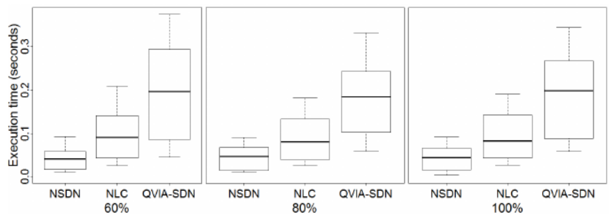
7.3.5. Mean runtime to allocate VI requests. Figures 8(a) and 8(b) shown the mean runtime to allocate VIs. As expected, the introduction of SDN requirements increased the number of constraints on LP, and consequently the runtime. Despite this fact, the average runtime is in order of few seconds for most cases.

7.4. Key observations

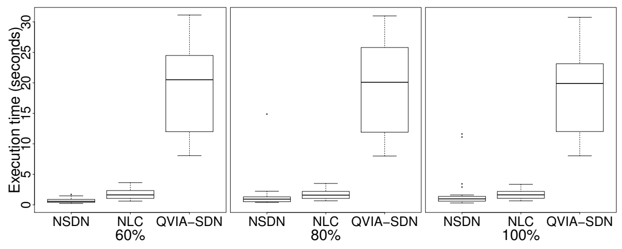
An SDN-aware allocation mechanism can increase the acceptance ratio without decreasing the QoS. As indicated by Figures 3 and 5, even allocating more VI requests, QVIA-SDN still composes VIs with lower internal latency values.

QoS requirements can be provisioned without decreasing the revenue-to-cost ratio. The cloud providers' objectives (discussed in Section 4.3) represented by Figures 4 and 7 can be achieved without overloading the data center as indicated by the fragmentation metric (Figure 6).

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



(a) $k = 4$.



(b) $k = 8$.

Figure 8. Mean runtime to allocate VI requests.

cloud data center homogeneity, a suitable solution can be found without analyzing all servers, switches, and paths.

8. Conclusion

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 formulate 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 modelling 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 [23], while a second line indicates an actual and practical implementation on private cloud frameworks.

Acknowledgments

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

References

[1] S. S. Manvi and G. K. Shyam, "Resource management for infrastructure as a service (iaas) in cloud computing: A survey," *Journal of Network and Computer Applications*, vol. 41, pp. 424 – 440, 2014.

[2] M. Rost, C. Fuerst, and S. Schmid, "Beyond the stars: Revisiting virtual cluster embeddings," in *In Proc. ACM SIGCOMM Computer Communication Review (CCR*, 2015.

[3] T. Truong Huu, G. Koslovski, F. Anhalt, J. Montagnat, and P. Vicat-Blanc Primet, "Joint elastic cloud and virtual network framework for application performance-cost optimization," *Journal of Grid Computing*, vol. 9, no. 1, pp. 27–47, 2011.

[4] K. He, E. Rozner, K. Agarwal, Y. J. Gu, W. Felter, J. Carter, and A. Akella, "Ac/dc tcp: Virtual congestion control enforcement for datacenter networks," in *Proceedings of ACM SIGCOMM 2016 Conference*. New York, NY, USA: ACM, 2016, pp. 244–257.

[5] B. Cronkite-Ratcliff, A. Bergman, S. Vargaftik, M. Ravi, N. McKeown, I. Abraham, and I. Keslassy, "Virtualized congestion control," in *Proceedings of the 2016 ACM SIGCOMM Conference*. New York, NY, USA: ACM, 2016, pp. 230–243.

[6] 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*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[7] R. Sherwood, G. Gibb, K. kiong Yap, M. Casado, N. Mckeown, and G. Parulkar, "Flowvisor: A network virtualization layer," Tech. Rep., 2009.

[8] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Can the production network be the testbed?" in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*. USENIX, 2010, pp. 1–6.

[9] N. M. M. K. Chowdhury and R. Boutaba, "Network virtualization: State of the art and research challenges," *Comm. Mag.*, vol. 47, no. 7, pp. 20–26, Jul. 2009.

[10] A. Fischer, J. Botero, M. Till Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 4, pp. 1888–1906, Fourth 2013.

[11] M. Chowdhury, M. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, Feb 2012.

[12] G. De S Cavalcanti, 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*, April 2014, pp. 1–8.

[13] 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*, Nov 2014, pp. 412–417.

[14] F. Tao, B. Jun, and W. Ke, "Allocation and scheduling of network resource for multiple control applications in sdn," *Communications, China*, vol. 12, no. 6, pp. 85–95, June 2015.

[15] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *Internet Computing, IEEE*, vol. 17, no. 2, pp. 20–27, March 2013.

[16] N. Kang, Z. Liu, J. Rexford, and D. Walker, "Optimizing the "one big switch" abstraction in software-defined networks," in *Proceedings of the 9th ACM CoNEXT '13*. New York, NY, USA: ACM, 2013, pp. 13–24.

[17] M. Demirci and M. Ammar, "Design and analysis of techniques for mapping virtual networks to software-defined network substrates," *Computer Communications*, vol. 45, pp. 1 – 10, 2014.

[18] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008.

[19] V. Persico, P. Marchetta, A. Botta, and A. Pescape, "Measuring network throughput in the cloud: The case of amazon {EC2}," *Computer Networks*, vol. 93, Part 3, pp. 408 – 422, 2015, cloud Networking and Communications {II}.

[20] N. M. K. Chowdhury and R. Boutaba, "Network Virtualization: State of the Art and Research Challenges," *Communications Magazine, IEEE*, vol. 47, no. 7, pp. 20–26, 2009.

[21] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 242–253, Aug. 2011.

[22] 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 *Proceedings of the 6th International COnference*, ser. Co-NEXT '10. New York, NY, USA: ACM, 2010, pp. 15:1–15:12.

[23] L. Popa, A. Krishnamurthy, S. Ratnasamy, and I. Stoica, "Faircloud: Sharing the network in cloud computing," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, ser. HotNets-X. New York, NY, USA: ACM, 2011, pp. 22:1–22:6.

[24] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, G. Parulkar, E. Salvadori, and B. Snow, "Openvirtex: Make your virtual sdns programmable," in *Proceedings of the 3rd Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. ACM, 2014, pp. 25–30.

[25] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, 2008.

[26] 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*, May 2011, pp. 153–160.

[27] W. Stallings, *Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud*, 1st ed. Addison-Wesley Professional, 2015.

[28] R. Mijumbi, J. Serrat, J.-L. Gorricho, and R. Boutaba, "A path generation approach to embedding of virtual networks." *CoRR*, vol. abs/1509.07684, 2015.

[29] D.-S. Chen, R. G. Batson, and Y. Dang, *Applied integer programming: modeling and solution*. John Wiley & Sons, 2010.

[30] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, pp. 1–53, 2014.