Virtual Network Function Embedding in Real Cloud Environments

Paolo Bellavista[1,2], Franco Callegati[1,3], Walter Cerroni[1,3], Chiara Contoli[3], Antonio Corradi[1,2], Luca Foschini[1,2], Alessandro Pernafini[1], Giuliano Santandrea[1]

[1]Center for Industrial Research on ICT (CIRI ICT) – University of Bologna
Viale del Risorgimento, 2 - 40136 Bologna - ITALY
Email: {alessandro.pernafini, giuliano.santandrea2}@unibo.it

[2]Department of Computer Science and Engineering (DISI) – University of Bologna
Viale del Risorgimento, 2 - 40136 Bologna - ITALY
Ph.: +39-051-20-93033; Fax: +39-051-20-93869
Email: {paolo.bellavista, antonio.corradi, luca.foschini}@unibo.it

[3]Department of Electrical, Electronic, and Information Engineering “G. Marconi” (DEI)
University of Bologna - Via Venezia, 52 - 47521 Cesena - ITALY,
Ph.: +39-0547-3-39209; Fax: +39-0547-3-39208
Email: {franco.callegati, walter.cerroni, chiara.contoli}@unibo.it

Abstract

Future network architectures will be completely reshaped by the emerging Network Function Virtualization (NFV) paradigm, and telco operators will likely deploy flexible infrastructures based on the cloud, offering programmable connectivity services and computing/storage facilities in the form of Virtual Data Centers (VDCs). This paper introduces and discusses some challenging technical issues associated with this promising approach, with a special focus on the exploitation of industry-relevant cloud platforms, such as OpenStack, and on network-aware optimal placement problem of entire VDCs, taking into account multiple virtual (Virtual Machines - VMs, virtual networks, …) and physical (host capacities, network topology and capacity, …) resources and constraints. The proposed placement, computed for real production OpenStack deployments, not only satisfies the predicted communication for all VDCs deployed atop the same physical data center, but also accounts for the (real) computing overhead due to intense communication load of VMs co-located on the same physical host, such as in the case of Virtual Network Functions (VNF) embedding, typically neglected by similar existing works.

Keywords: Cloud Management; Cloud Networking; Network Function Virtualization (NFV); Virtual Data Centers (VDC); VDC placement; Open Stack; Neutron.

1. Introduction

Cloud computing architectures enable elastic resource utilization and dynamic resource reconfiguration, by allowing both cloud users to reduce business costs by outsourcing IT services, and cloud providers to gain economical revenues from underutilized IT resources. One of the still open core management problems of current cloud systems at the infrastructure level is the most efficient *placement*, and dynamic relocation, of virtual resources over physical ones [1]. This includes not only Virtual Machine (VM) allocation, namely *VM placement*, but also the notable case of deploying entire Virtual Data Centers (VDCs), namely *VDC placement*, including multiple VMs and all their interconnection requirements.

Meanwhile, Network Function Virtualization (NFV) is an emerging paradigm [2], which promises to change the landscape of edge networks, where most of the functions required to analyze, classify, condition and secure data traffic are located and implemented by vendor-dependent middleboxes [3].

Operators typically apply an ordered set of network functions, i.e. a service chain, to a given traffic class or to a given user (or class of users). In the case of network functions implemented by middleboxes, this requires specific configuration and management task, reconfiguring physical switches, routers, network connectivity, etc. Virtualization brings the immediate advantage to make network function deployment independent of the underlying hardware, thus simplifying planning and management, given that, to date, the most typical approach to service chaining is static.

Nonetheless this is just the most straightforward advantage: NFV provides a novel degree of flexibility since the virtual environment makes possible to dynamically define and selectively apply instances of Virtual Network Functions (VNFs) that can be placed where they are actually needed, dynamically migrated, duplicated, or deleted according to the emerging user and/or network requirements [4]. As such, the service chain dedicated to a given customer or set of customers may be dynamic, changing over a rather small time scale, driven by external commands and/or by the changing network conditions.

A possible approach to this novel paradigm is to implement dynamic infrastructures at the edge of the network dedicated to specific service chains for specific clusters of users. These infrastructures are the VDCs this paper refers to in general. Given their final purpose, the performance of the interconnection between VMs in the VDC play a crucial role, as does the configurability of such connectivity. In state-of-the-art platforms the virtual network of the VDC can be controlled by means of Software Defined Networking (SDN) solutions, which represent the other building block boosting the aforementioned innovation. Owing to SDN, the virtual network of the VDC may be reshaped to adapt to changing service requirements. As a consequence, this work assumes that NFV and SDN are able to foster new forms of network virtualization of components and functions that will definitely change the shape of future network architectures.

This paper introduces and discusses some of the most challenging technical issues in this area: it presents the problem of integrated management of virtual IT and connectivity resources, describes possible relevant performance issues and bottlenecks, and proposes a new network-aware VDC optimal placement problem. The remainder of our paper contributes to the field by presenting results regarding the virtual networking performance within Data Centers (DCs) operating multiple VDCs owned by multiple users (i.e., multiple tenants). To support such results it provides thorough insights on how OpenStack, a popular open-source cloud platform, implements multi-tenant network virtualization, and discusses the related primary performance issues. The manuscript reports also the practical experience of the authors while assessing the proposed solution, by including a set of experimental results collected on a real OpenStack deployment. Finally, it uses those results in an original way to tune the network-aware VDC optimal placement problem [5] by adding new constrains on physical resources (CPU and memory) to account for the effects of intense virtual network usage for local communication over computing resources.

2. NFV in open source cloud systems

This section creates the common ground necessary for a better understanding of our work, by also sketching some related contributions.

2.1. Technology scenario and associated challenges

NFV is still in its infancy and its embedding in open-source technologies seems to be a significantly accelerating factor. A possible and industrially effective approach to NFV is to deploy network functions in VMs inside a suitable Data Center (DC) infrastructure, where network nodes and functions are deployed on standard, general-purpose, off-the-shelf hardware that can be kept inexpensive but powerful enough to provide the required Quality of Service. Whole network sections, such as the access where user-oriented functions are concentrated today, can become DC-like and their virtualized network functions (even operated by diverse players) can be dynamically deployed, cloned, migrated, modified, and removed according to customer usage and mobility needs. Then, the access can be interconnected through simpler, less hierarchical, and high-performance core networks that provide inter-DC connectivity service.

It is easy to understand that such a radical paradigm shift may completely reshape the value chains in the entire telco world, thanks to the many associated potential advantages, e.g., increased flexibility of management operations, improved state encapsulation, and the coexistence of sets of virtual network functions dedicated to different, isolated customers. In particular, such coexistence should allow the efficient sharing of a physical infrastructure, thus reducing operational costs in multi-tenant deployments.

The technical challenges of this innovative scenario mainly relate to its efficiency in supporting the operations of virtual network functions. The SDN paradigm provides the tools to automate virtual networking configuration, manage faults, recoveries, reconfigurations, etc. Therefore, these new technological challenges can synergically exploit NFV and SDN to provide the best possible and dynamic matching between logical and physical infrastructures. In more practical terms, that implies the capability of effectively managing:

- instantiation, allocation, and live migration of VMs within the virtualization infrastructure, including VMs that provide specific virtualized network functions;
- configuration of the virtual infrastructure networking environment to implement many isolated virtual networks distributed over a given set of potentially remote physical resources.

In other words, all VMs, hosting either computing or networking components, have to be managed in a holistic way to allow the orchestration of related resources within the DC infrastructure according to operation/optimization goals of the operator. This brings forward novel challenges in terms of components integration, management, and performance, especially for VDC placement. In fact, VDC placement requires not only placing multiple VMs, but also taking care of intertwining them through virtual networking resources across complex DC systems, including also external constraints due to physical and virtual network topologies, routing schemes, and traffic patterns (typically hard to predict). Moreover, modern cloud environments are generally used and shared by several tenants (i.e., *multi-tenant*) and management support is required to harmonize their different resource requirements and to grant proper isolation of all co-existing VDCs. The goal is to optimally exploit all available physical computing and networking DC resources, while avoiding severe performance degradation due to aggregated resource consumption of co-located VDCs deployed atop of it.

Achieving the aforementioned objectives requires:

- a DC platform providing integrated computing and network resources management;
- an in-depth understanding of how the virtual networking environment in the DC is implemented and managed;
- algorithms to design VDC placement considering both single node resource constraints and communication requirements/patterns.

Up to now, most research activities addressed VDCs by considering only resource constraints at single physical nodes, while we explicitly consider also the effects of network requirements and constraints in terms of communications between all VMs that belong to the same VDC [5]. In the past two years, some solid works have addressed this specific optimization problem, typically referred to as *network-aware VDC* (and *VM*) *placement*, but the applicability of these results in real industrial cloud environments is still widely unexplored. In fact, to the best of our knowledge, there are limited in-depth studies about the actual performance limits of NFV and network virtualization in cloud platforms, such as those due to heavy inter-VM communications, belonging to either the same or different VDCs and consolidated over the same physical server. Although some initial results in this direction were presented with reference to the open-source and widely diffused cloud platform OpenStack[1] in [6] and [7], there is still much room for research in this area.

2.2. Related work

NFV-enabled VDC placement involves many research areas, spanning from placement

[1] OpenStack Cloud Project. Available: http://www.openstack.org/

optimization problems and performance analysis to network and systems management. For the sake of briefness and with no ambition of exhaustiveness, in the following we focus on two main areas: management solutions for network-aware resource placement in DCs and NFV issues in virtualized cloud environments.

The placement of virtual resources in DCs has been widely addressed, but most literature approached placement by only considering single node resource constraints, without taking into account the practical effects of multi-tenants' different network requirements. A useful survey on resource management in cloud environments can be found in [8]. Oktopus [9] strives at allocating resources to virtual clusters required by tenants, by taking into account the virtual network topology that interconnects the required VMs, in order to find a tradeoff between SLA, costs and provider revenue. A work based on a Stochastic Bin Packing problem describes a VM consolidation solution that is aware of VM bandwidth requests [10]. Another proposal strives at finding affine VMs (similarity based on network traffic patterns), grouping them into a single unit of allocation, and deploying units into physical hosts based on a heuristic bin packing algorithm [11]. Another approach, instead, focuses on optimizing multiple resource utilization at the same time [12]: the proposal addresses the VM placement problem by combining bin packing to optimize node resource exploitation and quadratic assignment to minimize network traffic. A VM migration problem is addressed by S-CORE, which strives at migrating and consolidating VMs that heavily communicate with each other with the main goal of reducing traffic over the topology links [13]. A solution closer to our proposal defines an optimal VM placement problem with the goal of improving network scalability [14]: the proposed approximate algorithm to solve the problem, which is demonstrated to be NP-hard, aims at localizing large chunks of traffic to reduce traffic load at high-level switches. Similarly to our solution, the provided heuristic is based on a two-phase algorithm that first tries to group VMs to cluster and then maps VMs to physical hosts; however, authors do not consider the load on each host due to the virtual network infrastructure to enable local communications between VMs placed on the same host. Finally, another work presents a VM placement optimization problem with the main goal of improving the efficiency of VM communications towards the DC entry point [15]; also in this case the feasible solution is to provide approximated algorithms, applicable only to the notable case of tree-based network topology (see Section 4).

Except for some surveys on VNE and on architectural/technical solutions for NFV and SDN issues in cloud environments [1] [16], very limited work has been done so far to assess the actual performance limitations of network virtualization in real cloud platforms. An experimental evaluation of network virtualization overhead over Amazon EC2 shows that network performance might degrade significantly due to mutual interference between CPU utilization for computing and networking [17]. Other experimental works focus on similar limitations of OpenStack, by providing some preliminary results on degradation due to consolidation of highly network-intensive VMs on the same physical server [6] [7].

The aforementioned solutions separately focus either on partial aspects of network-aware placement, or on assessment of cloud networking performance. In contrast, our work jointly considers network constraints at both host and DC levels, by proposing an original network-aware VDC placement optimization solution that takes into account our practical experience on real OpenStack deployments.

3. OpenStack as a platform for cloud deployment of virtual network functions

The significant advantages in terms of flexibility and rapid service deployment offered by both computing and network resource virtualization have fostered the development of new open-source cloud computing platforms implementing the IaaS paradigm, such as OpenStack. OpenStack provides cloud managers with a powerful set of tools and a flexible dashboard that allow controlling a cluster of servers with different hypervisors and managing the required storage and virtual network facilities. A typical OpenStack deployment consists of a number of physical nodes, including:

- a *controller node*, to manage the whole cloud platform;
- a *network node*, hosting the cloud network services (e.g., DHCP, NAT, firewall, etc.) and providing the VMs with external connectivity;
- a number of *compute nodes*, where the VMs are actually running;
- a number of *storage nodes*, for storing data and VM images.

In most recent versions, an OpenStack component named *Neutron* has been specifically dedicated to deal with network service management: Neutron decouples *network abstractions*, such as layer 2 (L2) networks and layer 3 (L3) subnets, from their actual implementation, e.g., physical and virtual switches/routers. A centralized Neutron server stores and orchestrates all network-related information, whereas Neutron agents, running on the network node and on any compute node, implement the virtual network infrastructure in a distributed way. This allows Neutron to manage multi-tenant networks over multiple compute nodes and to provide transparent VM mobility within the VDC.

The OpenStack dashboard can be used also by cloud customers to quickly and transparently instantiate resources within a VDC infrastructure. In particular, an OpenStack user, belonging to a given tenant, is allowed to create new L2 networks, define one or more L3 subnets on top of them, and boot a number of new VM instances, by specifying the subnet (or subnets) they have to be connected to. If needed, a floating IP obtained from the cloud provider domain can be assigned to the tenant and used as an external address to ensure VMs reachability from outside the cloud. All the required virtual network appliances (such as routers and NATs providing global connectivity, firewalls, load balancers, etc.) can be either implemented directly in the cloud platform by means of *Linux containers* and *namespaces* (typically inside the network node), or deployed as ad-hoc VMs by the tenant itself in its virtual infrastructure. This is a very powerful and flexible feature available in OpenStack, making it a valuable open-source candidate for effective and vendor-independent VDC and NFV deployment.

While the use of VMs is a well-established solution to guarantee complete control of each tenant on its own virtual hosts, one of the most challenging issues for cloud providers is to achieve robust isolation also among different tenants' virtual networks. To this purpose, Neutron adopts a set of cross-layer software tools (including VLANs, network namespaces, OpenFlow rules, packet filtering rules, and tunneling protocols) and orchestrates them in a unified and effective way. More specifically, the OpenStack virtual network infrastructure consists of multiple virtual bridges running in each node:

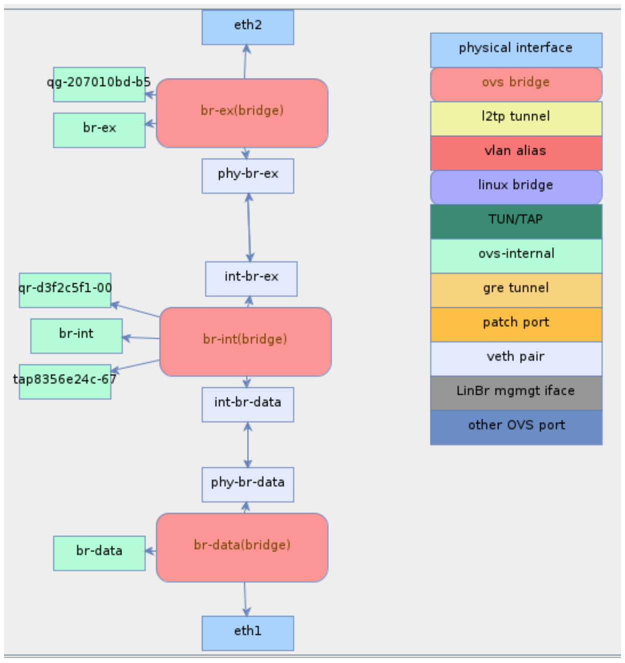
- each VM is connected to its L2 networks by means of a dedicated Linux Bridge, the native kernel-level software bridging tool;
- an integration bridge, named *br-int* and implemented with SDN-capable Open vSwitch (OVS) software technology, is used to interconnect all Linux Bridges and VMs to the cloud network infrastructure;
- an additional OVS bridge is used to connect br-int to each physical network the node is attached to.

L2 virtualization and multi-tenant isolation on the physical data network can be implemented by using either VLANs or overlay tunneling technologies, such as Virtual eXtensible LAN (VXLAN) or Generic Routing Encapsulation (GRE), which allow extending local virtual networks to remote DCs *[16]*. Whatever virtualization technology is adopted, its virtual networks must be mapped into the VLANs used internally by Neutron to achieve isolation. This is performed by taking advantage of OVS programmability through OpenFlow mapping rules. In principle, the VMs could be connected directly to the integration bridge. However, some security features are currently implemented by Neutron by applying the native kernel filtering functions (*netfilter*) to bridged interfaces, and this approach can work only with Linux Bridges: for this reason an additional Linux Bridge is needed as an intermediate element to interconnect the VM interface to the integration bridge.

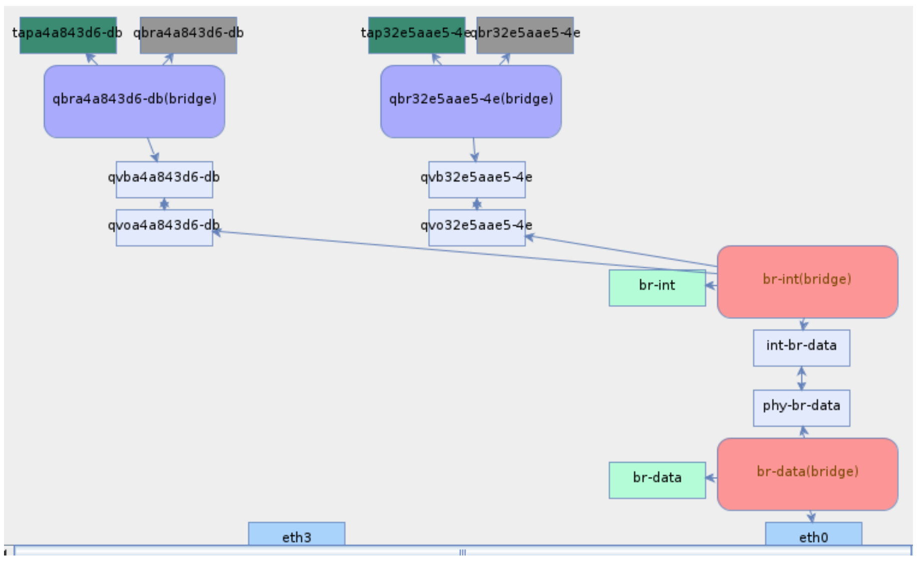
The resulting OpenStack network infrastructure consists of a non-trivial set of interconnected physical and virtual switching and routing systems. The complexity of the software components running inside a network node and a compute node is illustrated in Fig. 1 for a simple case of two VMs running in the same compute node. All these virtual network components might relevantly affect the physical server processing workload, in particular in case of network-intensive applications, as shown by our experimental results. Therefore, these performance degradations should be carefully considered and taken into account when modeling the network-aware VDC placement optimization problem as discussed in the next section.

4. Network-aware cloud management support

Efficient network-aware VDC placement is crucial for effective and feasible NFV solutions, in particular when considering the on-demand provision of multi-tenant VDCs. We propose a solution approach for the optimal network-aware VDC placement problem called Min-Cut Ratio-aware VM Placement for effective NFV (MCRVMP-N). This approach extends our previous work **[5]** by taking into account realistic constraints of additional load due to local communication in cloud deployments. These constraints were derived from an extensive set of experiments on a real OpenStack deployment, as partially described in Section 5.1. Since the MCRVMP-N optimization problem is NP-hard, we also propose an original heuristic to solve it in a reasonable and practically feasible time.



(a) Network node



(b) Compute node

Fig. 1 Networking elements in OpenStack network node (a) and compute node running two VMs (b). The Open vSwitch bridges (red boxes) are interconnected by virtual Ethernet pairs (grey boxes). Inside the network node, *br-data* and *br-ex* are attached to their respective physical interfaces (*eth1* and *eth2*, connected to the internal and external physical networks, respectively), whereas the compute node is connected to the internal physical network only (via *br-data* and *eth0*). Inside the compute node, two Linux Bridges (light-blue boxes) are attached to the VM tap interfaces (green boxes) and connected by virtual Ethernet pairs to the Open vSwitch integration bridge (*br-int*). The screenshots have been obtained from a graphical tool developed by one of the authors and available for download at https://sites.google.com/site/showmynetworkstate.

Before detailing MCRVMP-N let us provide some needed background about typical DC network topologies, usually based on three-tier architectures [18]. The lowest *access tier* contains physical hosts connected to access switches. The intermediate *aggregation tier* contains aggregation switches to connect access switches, with the responsibility of aggregating traffic to/from the underlying layer. Aggregation switches are connected through core switches contained in the highest *core tier*, together with gateways to enable external communications. Based on a three-tier architecture, three topologies are emerging as standard-de-facto solutions: tree, fat-tree, and VL2 (see Fig. 2). The main advantage of tree topologies is the reduced economical cost thanks to wiring simplicity. Unfortunately, they suffer from scalability bottlenecks because the links closer to the tree root have to carry traffic coming from big DC portions. Moreover, they are also weak in terms of reliability, i.e., if a link connecting two switches breaks down. On the contrary, fat-tree [19] and VL2 [20] topologies are more expensive, being based on bipartite graphs to grant higher scalability and reliability (higher number of paths for traffic routing between any pair of physical hosts).

In this context, we claim the need for network-aware VDC placement solutions able to take into account both local resource constraints (such as CPU and memory) and network constraints, in order to find placement configurations able to absorb unpredicted traffic bursts. More in particular, network constraints are introduced to limit the load due to local communications of multiple VMs deployed on a same host that was not considered in our previous work [5]. From a more technical perspective, to efficiently face time-varying traffic demands and traffic, our goal is minimizing the worst case Cut Load Ratio (CLR), so to limit packet drops and VM relocations. In our approach, we consider critical cuts, namely network cuts that partition the set of hosts into two non-empty connected subsets, which can become bottlenecks for the traffic

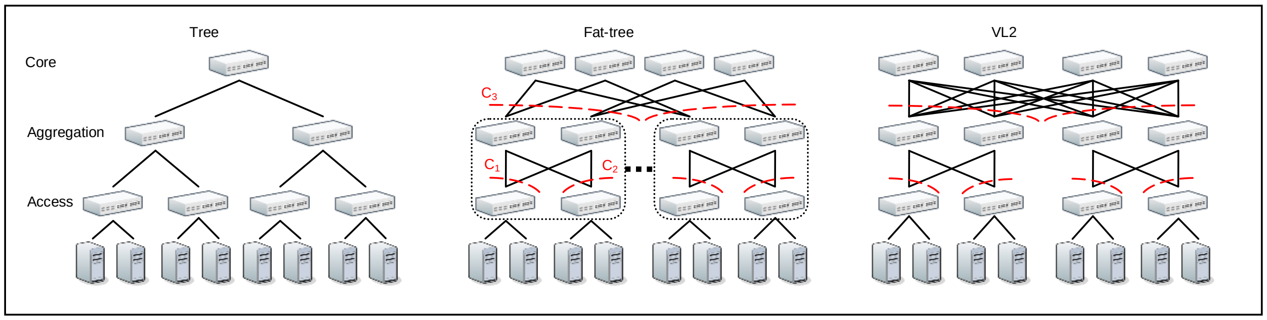


Fig. 2 Common data center network topologies

demands between VMs placed in different sides of the cut. Whenever a critical cut reaches a load ratio close to 1, it means that it could sustain only few variations in the traffic demands before malfunctioning, and starting to drop packets. In a tree network topology, each critical cut corresponds to a single network link; hence, the failure of one link leads to the partition of the network into two separate subsets of connected hosts. The cut analysis is more complicated for topologies like fat-tree and VL2, which are based on bipartite graphs. Without loss of generality, in our work we use the tree network topology because it is always possible to transform more complex topologies into equivalent tree topologies, as demonstrated in our previous work [5]. In other words, we solve the equivalent placement problem for the tree network topology and we can apply the obtained placement solution, as well as the load values on the critical cuts, to the initial complex topology (e.g., fat-tree and VL2).

Our original MCRVMP-N approach is designed for high variability of traffic demands and tries to find placement solutions resilient to traffic bursts in deployed VMs, by minimizing the maximum load ratio over all network cuts, while respecting constraints on host CPU/memory and network cuts. In particular, MCRVMP-N tries to place VMs to hosts. We consider a set of n host $H=\{h_i\}_{i=1,\ldots n}$ and a set of m VMs $=\{vm_j\}_{j=1,\ldots m}$. We describe each VM vm_j by resource requirements $\langle CPU_{REQ}, MEM_{REQ}, NET_{REQ}\rangle$, namely, CPU and memory computing resources, and networking requirements in terms of total bandwidth for local communications between the VM and any other VM deployed on the same host. Each host h_i, instead, is characterized by resource capacities $\langle CPU_{CAP}, MEM_{CAP}, NET_{CAP}\rangle$, namely, overall CPU and memory available at the physical host, and network capacity available for local communications between VMs deployed at the same physical host. The network capacity is experimentally calculated, and represents the threshold, above which local communications incur performance degradation. Our approach originality also stems from the consideration of average traffic rates between VMs to calculate traffic rate flowing through network cuts, but, incrementally to [5], it also accounts for an additional constraint on local network capacity due to the CPU overhead introduced by virtual networking components (for intensively communicating VMs co-located at the same host). The average traffic rates between VMs is represented by a traffic matrix T, where the element t_{ij} is the traffic rate from vm_i to vm_j. Finally, we used X, an m*n matrix of binary variables to represent VM placement; the element x_{ij} is 1 if vm_i is placed on host h_j, 0 otherwise.

As already stated, each cut partitions the set of hosts in two disjoint subsets, namely H1 and H2, where H1 is always the sub-tree that does not contain the original tree root. We define CLR for a specific cut as the ratio between the traffic load from H1 to H2 divided by the capacity of the network cut, and the same ratio in the opposite direction. In a valid placement, the aggregated traffic flowing from H1 to H2 must be lower than the cut capacity (bi-directionally). For this to hold, CLR must be lower than 1 for each network cut. Finally, to find the optimal placement, we define MCRVMP-N goal as the minimization of the worst case CLR. The cut matrix C is used to express all the critical cuts in the topology. Each row represents a network cut and each column a host; the element c_{di} is 1 if h_i belongs to H_1 when the dth cut is considered. It is very simple, for a tree topology, to generate the C matrix. In fact, for each row, it only requires to remove the corresponding link and express the row with the two host partitions H_1 and H_2. By using the cut matrix C, the total traffic rate from H_1 to H_2 (respectively, H_2 to H_1) over the dth cut is the dth-element of the diagonal of the matrix $[C * X^T * T * X * (1\text{-}C)^T]$ (respectively, $[(1\text{-}C) * X^T * T * X * C^T]$). This traffic rate only considers the traffic among VMs; we also consider the traffic from/to the gateway, situated at the tree root, but, for the sake of readability, it is not shown in the following model.

$$\min\left(\max_{c\,\in 1\ldots 2*N_{cut}} \mathrm{CLR_c}\right) \qquad (1)$$

$$\sum_i \mathrm{vm_i.CPU_{REQ}} * \mathrm{x_{ij}} \le \mathrm{h_j.CPU_{CAP}} \quad \forall \mathrm{j} \qquad (2)$$

$$\sum_i \mathrm{vm_i.MEM_{REQ}} * \mathrm{x_{ij}} \le \mathrm{h_j.MEM_{CAP}} \quad \forall \mathrm{j} \qquad (3)$$

$$\sum_i \mathrm{vm_i.NET_{REQ}} * \mathrm{x_{ij}} \le \mathrm{h_j.NET_{CAP}} \quad \forall \mathrm{j} \qquad (4)$$

$$\mathrm{CLR_c} = \begin{cases} \dfrac{[C * X^T * T * X * (1 - C)^T]_{cc}}{CAP_c}, \forall c \in \{1, \ldots, N_{CUT}\} & (5) \\ \dfrac{[(1 - C) * X^T * T * X * C^T]_{cc}}{CAP_c}, \forall c \in \{N_{CUT} + 1, \ldots, 2 * N_{CUT}\} & (6) \end{cases}$$

$$\mathrm{CLR_c} \le 1 \;\forall \mathrm{c} \in \{1, \ldots, 2 * \mathrm{N_{CUT}}\} \qquad (7)$$

$$\sum_{j} x_{ij} = 1 \quad \forall i \tag{8}$$

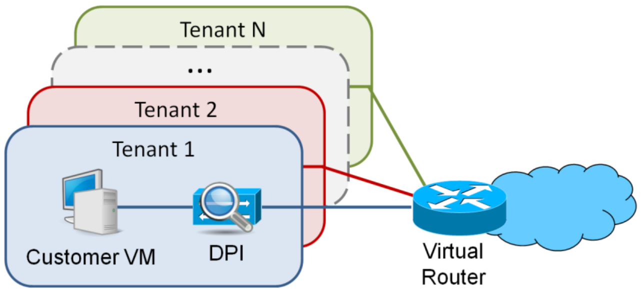
$$x_{ij} \in \{0, 1\} \quad \forall i, \forall j \tag{9}$$

Formula (1) represents MCRVMP-N goal that is to minimize the worst case cut load ratio. Formulas (2), (3), and (4) express CPU, memory, and local network constraints at each single host. Formulas (5) and (6) enforces the aggregate traffic flowing on each network cut, while formulas (7) establish feasible solutions from the network point-of-view. Finally, formulas (8) avoid the same VM to be placed on multiple hosts.

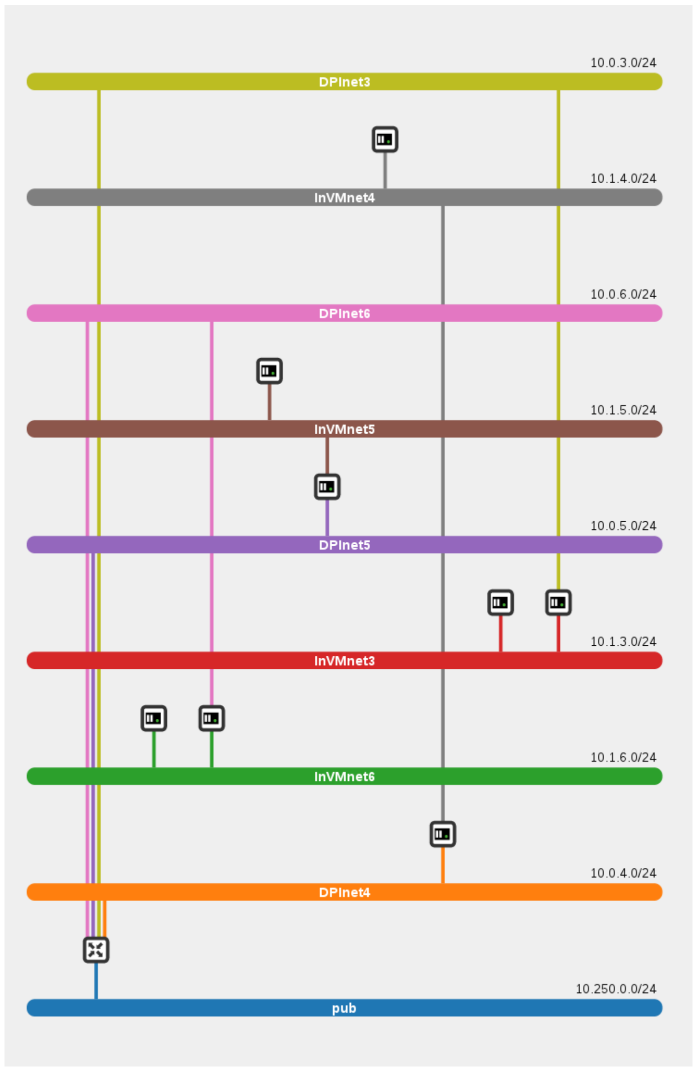
MCRVMP-N is an NP-hard problem that can be optimally solved only for small problem instances. In order to apply it to a real medium size OpenStack DC scenario, we used a heuristic called 2-Phase Connected Component-based Recursive Split (2PCCRS). 2PCCRS exploits the idea that typical cloud scenarios consist of groups of VMs that participate to the same VDC and exchange data only within themselves or with the external gateway (we refer to VMs running a typical three-tier application); these VMs can be grouped into bigger entities called Connected Components (CCs). 2PCCRS processes the traffic matrix T and clusters VMs in $CC=\{cc_d\}_{d=1,\ldots,t}$ accordingly. Each cc_d is associated with a vector of resource requirements $\langle CPU_{TOT}, MEM_{TOT}, NET_{TOT}, IN_{TOT}, OUT_{TOT}\rangle$, representing aggregated requirements in terms of CPU, memory, network bandwidth for local communications inside a same host, download and upload traffic with the gateway. CCs are built so that they do not exchange traffic among them, hence we only need to model the traffic with the gateway. The algorithm is recursive and adopts a hierarchical two-phase approach. The first phase, starting from the tree root, considers CCs (namely, VDCs) as single VMs and tries to recursively place them in a sub-tree associated to a switch. At the end of the first phase, if a CC has not been placed due to resource capacity constraints, it means that its VMs can be placed on every host in the second phase. The second phase, also starting from the root, splits the CCs to recursively place VMs on physical hosts, by solving MCRVMP-N problems associated to the sub-tree where CCs were placed in the previous phase. The problem instances to solve in the second phase are usually small, thus making the 2PCCRS heuristic suitable also for realistic cloud environments.

5. Performance evaluation

To assess the effectiveness and the overhead cost of our solution, we ran several experiments over two different directions: the former consisting of real on-the-field experiments to measure performance degradation caused by the OpenStack virtual network complexity in case of multiple tenants simultaneously active in the same node, the latter using the results obtained by these experiments to tune our MCRVMP-N optimization for OpenStack-based cloud infrastructure deployments.



(a)



(b)

Fig. 3 Multi-tenant NFV scenario tested on (a) the OpenStack platform and (b) a corresponding virtual network view from the OpenStack platform dashboard. Each tenant slice includes a VM connected to an internal network (InVMnet*i*) and a second VM performing DPI and packet forwarding between InVMnet*i* and DPInet*i*. The virtual router in the bottom-left corner provides connectivity between each DPInet*i* and the public network.

5.1 Multi-tenancy overhead on network performance

In order to quantitatively assess the impact of multi-tenancy on network performance, we set up a small test-bed including a controller node, a compute node and a network node. The compute node runs KVM, the native Linux VM Hypervisor, and is equipped with 8 GB of RAM and a quad-core processor enabled to hyper-threading, resulting in 8 virtual CPUs available to the Hypervisor.

The analyzed multi-tenant VDC scenario is inspired by a simple NFV case-study, illustrated in Fig. 3.a: each tenant operates a VDC service chain that consists of a customer-controlled VM followed by a deep packet inspection (DPI) virtual appliance, deployed by the cloud service operator as a separate VM with two network interfaces, running a traffic monitoring application based on the nDPI library[2]. All traffic exchanged by customers is subject to DPI processing, a common practice among Internet service providers to enforce network security and traffic policing. The NFV approach allows to flexibly deploy and customize the virtual DPI function based on the given tenant characteristics. Fig. 3.b shows the virtual network view from the OpenStack controller node dashboard when four tenants (VDCs) are simultaneously active. The *i*-th tenant slice is connected through an access network (DPInet*i*) to the virtual router, running in the network node within a dedicated network namespace, for external packet forwarding and NAT operations, while a VM performs DPI and packet forwarding between the tenant's internal network (InVMnet*i*) and DPInet*i*. This VM setup is not common in a traditional cloud computing environment, where VMs typically act as simple hosts connected as end-points to one or more virtual networks. However, the NFV paradigm is very likely going to change this approach and require that some VMs perform network functions involving packet forwarding among multiple network segments, as in our DPI example. Since this change of role breaks some of the filtering policies imposed by OpenStack security groups (such as those applied against MAC and IP spoofing), we had to reconfigure Neutron firewall and NAT services accordingly.

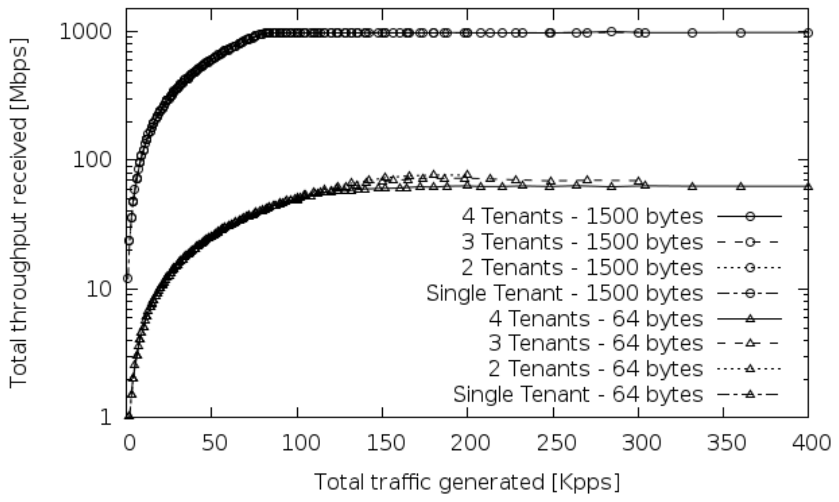
We assessed the performance of the OpenStack virtual network infrastructure by measuring, at an external destination, the packet rate and throughput received from traffic sources that run in each customer VM and generate packets at increasing rate, for different numbers of

[2] nDPI: Open and Extensible LGPLv3 Deep Packet Inspection Library. Available: http://www.ntop.org/products/ndpi/

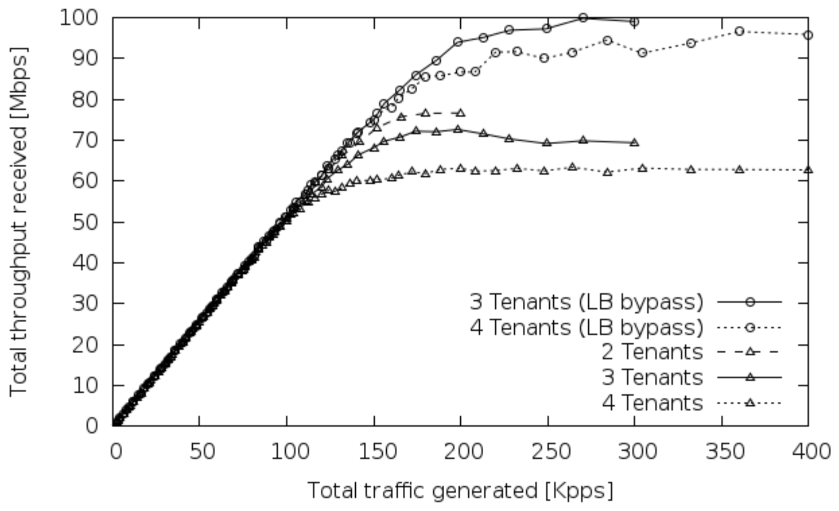
simultaneously active tenants. Each source generates UDP traffic ranging from 10^3 to 10^5 packets per second (pps), for both 64 and 1500-byte IP packet sizes. The *RUDE & CRUDE* tool[3] is used for traffic generation and measurement. All physical interfaces used in our experiments are Gigabit Ethernet network cards.

Fig. 4.a shows (in logarithmic scale) that the total throughput measured at the destination is proportional to the total packet rate generated by different numbers of simultaneously active tenants (1 to 4), for 1500 and 64-byte IP packet sizes.

[3] RUDE & CRUDE: Real-time UDP Data Emitter & Collector for RUDE. Available: http://sourceforge.net/projects/rude/



(a) Total throughput measured vs. total packet rate generated by 1 to 4 tenants for 1500 and 64-byte packet size. The curves are plotted in logarithmic scale due to the large difference in range values.



(b) Total throughput measured vs. total packet rate generated by 2 to 4 tenants for 64-byte packet size. Comparison between normal OpenStack mode and Linux Bridge bypass with 3 and 4 tenants.

Fig. 4 Multi-tenant virtual network performance in OpenStack compute node.

When the generated traffic is not sustainable anymore, two different saturating effects take place, depending on the packet size. The sharp saturation, obtained with 1500-byte packets when reaching 1 Gbps (corresponding to a total packet rate of about 80 Kpps), is clearly caused by the physical bandwidth limit imposed by the Gigabit Ethernet interfaces involved in the data transfer. A smoother saturation, at a much smaller throughput value, is experienced for 64-byte packets. This behavior is due to the inability of the hardware platform to cope with the packet processing workload in the complex OpenStack virtual network infrastructure. The interesting result is that the amount of such a performance degradation increases with the number of tenants (namely, VDCs and VMs therein), in other terms with the complexity of the virtual network.

To further analyze the detrimental effect of multi-tenancy and the role of the different virtual network components, Fig. 4.b compares the 64-byte total throughput obtained under normal OpenStack operations with the corresponding total throughput measured when the Linux Bridges attached to each VM are manually bypassed. The curves show that the presence of Linux Bridges in normal OpenStack mode is indeed causing performance degradation, especially when the workload is high (i.e., with 4 tenants), due to the increased packet processing burden on the physical computing resources.

The maximum throughput values measured at an external destination and obtained with our OpenStack test-bed for 64-byte packets are reported in Table 1, showing the performance penalty introduced, with respect to the single tenant case, when multiple tenants activate virtual networks according to the scenario of Fig. 3. We can observe that, in case of small packets and as long as the number of tenants does not exceed half the number of cores, the network performance degradation is negligible, while the penalty becomes significant with a growing number of tenants. These values have been used to tune the constraints that account for local virtual networking activities in our MCRVMP-N optimization problem.

Table 1 Effect of the number of tenants on the total throughput achieved with the OpenStack virtual network infrastructure on a quad-core processor with hyper-threading.

No. of tenants	No. of running VMs	Maximum throughput achieved	Relative penalty
1	2	77 Mbps	-
2	4	75 Mbps	2.5%
3	6	70 Mbps	9%
4	8	62 Mbps	20%

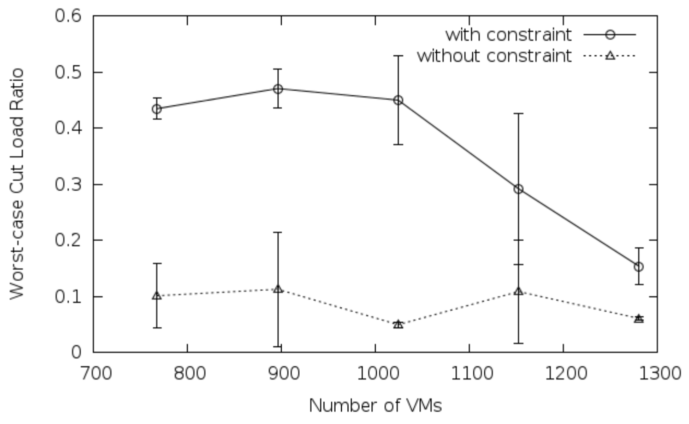
5.2 MCRVMP-N performance evaluation

The second set of experiments uses obtained performance penalty estimations (see Table 1) to tune the MCRVMP-N optimization problem that we applied to a wide network deployment by considering a fully balanced quaternary tree with 64 hosts. We do not consider unbalanced trees, because balanced trees are a more common topology for real-world datacenters [19]. In this case, we increment the number of VMs (from 12x to 20x the number of hosts), to compare heuristic scalability; as regards traffic matrix, CCs size follow a uniform distribution in [1; 22], while associated traffic pairs are generated with a probability of 0.75. All the following experimental results are associated with a data center made by a pool of homogeneous hosts having the same capacity for CPU and memory resources. In addition, all VMs have equal CPU and memory requirements and according to imposed capacity constraints, each physical host in the pool can accommodate the same number of VMs. The data center network is always a fully balanced tree with link capacity of 1 Gbps. We execute our heuristics on a Linux box with CPU Intel Core i7-3610QM @ 2.30GHz and 8 GB RAM, and we exploit IBM ILOG CPLEX as mixed integer mathematical solver to solve the intermediate steps of 2PCCRS. ILOG is always configured with pre-solve and parallel mode enabled. Finally, all the reported experimental results are average values over 10 different executions and we report standard deviation values to better assess the confidence of our results.

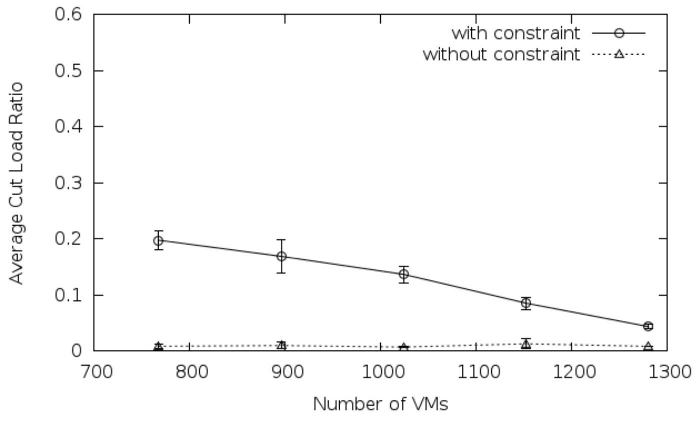
To assess the impact of the overhead due to local networking activities, we evaluated the same 2PCCRS heuristic algorithm with and without the new constraint (i.e., local networking constraint) on the networking capacity locally available at each node, which accounts for more CPU resources to guarantee highly performing networking at each local DC node. Fig. 5 reports the results of our experiments; Fig. 5a, 5.b, and 5.c show, respectively, the value obtained for the worst case maximum CLR, the average CLR (over all network cuts in the considered quaternary tree), and the 2PCCRS computation time for the MCRVMP-N problem. Continuous line represents the behavior with the new constraint, while the dashed one represents the behavior without it, when increasing the total number of VMs (reported on the x-axis).

The case without the local networking constraint always performs significantly better and leads to better VM placement solutions because more CPU resources are available; accordingly, a greater number of VMs can be placed under the same host, thus reducing both the worst case and average CLRs. Focusing on the case with local networking constraints, performances are significantly lower with worst case CLR up to 0.47, but found solutions are always far from network saturation and able to absorb unpredicted traffic bursts. For numbers of VMs higher than 1024, the gap between the two curves tends to diminish when the number of VMs increases. That behavior is due to the fact that, in order to compute CLR values on exactly the same network topology and to have enough computing resources to accommodate all VMs, we decided to scale physical hosts vertically by applying a multiplicative scaling factor. Consequently, for high enough values of number of VMs, the physical hosts have enough capacity to accommodate an entire VDC (namely all VMs within the same CC), thus resulting in a behavior that for very high values tends to the case without the local networking constraint.

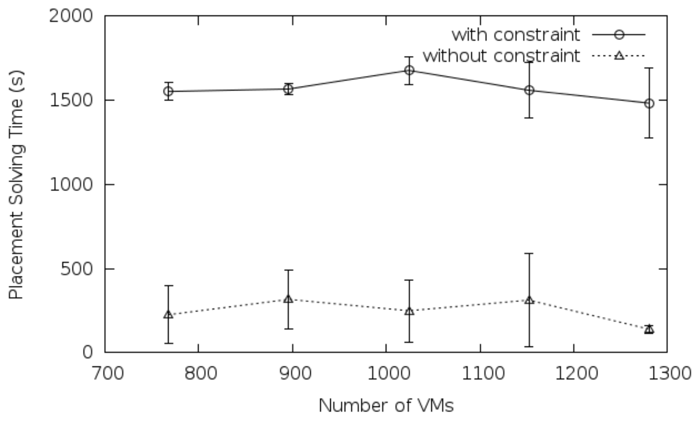
The 2PCCRS computation time is generally high, typically above 4 minutes in any case due to the usage of mathematical programming techniques (see Fig. 5.c). When the local network constraint is not applied, the time is smaller because the hosts can accommodate more VDCs. Adding the local networking constraint, instead, makes the problem more difficult due to the saturation of host capacities and the required time increases significantly. In this case, we limit maximum placement computation time to 1800 seconds because we found that this total solving time ensures a good tradeoff between computation time and solution quality. Let us stress again that these kinds of replacements are typically performed over long time periods, such as once per day, to reshuffle VDCs based on the communication history in the last period so to absorb unpredicted traffic bursts.



(a)



(b)



(c)

Fig. 5 (a) Worst case cut load value, (b) average cut load values, and (c) placement computation times for a topology of 64 hosts.

6. Conclusion

The NFV paradigm has the potential to completely reshape the future network architecture, fostering the evolution of both telco and cloud provider infrastructures toward a unified, cloud-based, multi-tenant, programmable platform, which offers flexible connectivity services and computing/storage facilities in the form of virtualized data centers. In this paper, we discussed how such a revolutionary approach must deal with the optimal deployment of virtualized nodes and networks while taking into account both processing and communication resource availability. A suitable solution for MCRVMP-N was devised by realistically considering physical and virtual network constraints imposed by a popular open-source cloud management platform such as OpenStack. The proposed approach allows the deployment of an optimal network-aware VDC placement strategy, which reduces the impact of the workload on the physical communication infrastructure while limiting the detrimental effect of multi-tenancy on the performance of the virtualized network infrastructure. This work represents a first insight to the many open issues that still need to be addressed before full-scale virtualization of networking services becomes a reality.

Acknowledgements

This research was supported in part by CIRI, Center for ICT technology transfer of the University of Bologna, funded by POR FESR Emilia-Romagna 2007-2013.

References

[1] A. Fischer, J. Botero, M. Till Beck, H. de Meer and X. Hesselbach, "Virtual Network Embedding: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888-1906, 2013.

[2] "ETSI Network Functions Virtualisation (NFV)," [Online]. Available: www.etsi.org.

[3] A. Manzalini, R. Minerva, F. Callegati, W. Cerroni and A. Campi, "Clouds of virtual machines in edge networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 63-70, July 2013.

[4] J. Soares, C. Goncalves, B. Parreira, P. Tavares, J. Carapinha, J. Barraca, R. Aguiar and S. Sargento, "Toward a telco cloud environment for service functions," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 98-106, February 2015.

[5] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz and E. Silvera, "A Stable Network-Aware VM Placement for Cloud Systems," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)*, 2012.

[6] F. Callegati, W. Cerroni, C. Contoli and G. Santandrea, "Performance of Network Virtualization in Cloud Computing Infrastructures: The OpenStack Case," in *Third International conference on cloud networking (Cloudnet 2014)*, Luxembourg, 2014.

[7] X. Ge, Y. Liu, D. H. Du, L. Zhang, H. Guan, J. Chen, Y. Zhao and X. Hu, "OpenANFV: Accelerating Network Function Virtualization with a Consolidated Framework in OpenStack," in *Proceedings of the 2014 ACM conference on SIGCOMM*, Chicago, USA, 2014.

[8] B. Jennings and R. Stadler, "Resource Management in Clouds: Survey and Research Challenges," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 567-619, 2015.

[9] H. Ballani, P. Costa, T. Karagiannis and A. Rowstron, "Towards Predictable Datacenter Networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, Toronto, Ontario, Canada, 2011.

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

[11] J. Chen, K. Chiew, D. Ye, L. Zhu and W. Chen, "Aaga: Affinity-aware grouping for allocation of virtual machines," in *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, 2013.

[12] J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang and S. Cheng, "Energy-saving virtual machine placement in cloud data centers," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, 2013.

[13] F. P. Tso, G. Hamilton, K. Oikonomou and D. P. Pezaros, "Implementing scalable, network-aware virtual machine migration for cloud data centers," in *Cloud Computing (CLOUD), 2013 IEEE Sixth International Conference on*, 2013.

[14] X. Meng, V. Pappas and L. Zhang, "Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement," in *INFOCOM, 2010 Proceedings IEEE*, 2010.

[15] R. Cohen, L. Lewin-Eytan, J. Seffi Naor and D. Raz, "Almost optimal virtual machine placement for traffic intense data centers," in *INFOCOM, 2013 Proceedings IEEE*, 2013.

[16] R. Jain and S. Paul, "Network Virtualization and Software Defined Networking for Cloud Computing: A Survey," *Communications Magazine, IEEE,* vol. 51, no. 11, pp. 24-31, November 2013.

[17] S. Ryan, W. Feng, W. Haiyang and L. Jiangchuan, "A deep investigation into network performance in virtual machine based," in *2014 IEEE Conference on Computer Communikations (INFOCOM 2014)*, Toronto, Canada, 2014.

[18] "Cisco Data Center Infrastructure 2.5 Design Guide," 2011.

[19] M. Al-Fares, A. Loukissas and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63-74, 2008.

[20] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang and S. Lu, "BCube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 63-74, 2009.