This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2413755, IEEE Transactions on Network and Science Management

Striking a Balance Between Traffic Engineering and Energy Efficiency in Virtual Machine Placement

Dallal Belabed, *Graduate Student Member, IEEE,* Stefano Secci, *Member, IEEE*, Guy Pujolle, *Senior Member, IEEE*, Deep Medhi, *Senior Member, IEEE*

***Abstract*—The increasing adoption of server virtualization has recently favored three key technology advances in data-center networking: the emergence at the hypervisor software level of virtual bridging functions, between virtual machines and the physical network; the possibility to dynamically migrate virtual machines across virtualization servers in the data-center network (DCN); a more efficient exploitation of the large path diversity by means of multipath forwarding protocols. In this paper, we investigate the impact of these novel features in DCN optimization by providing a comprehensive mathematical formulation and a repeated matching heuristic for its resolution. We show, in particular, how virtual bridging and multipath forwarding impact common DCN optimization goals, traffic engineering (TE) and energy efficiency (EE), and assess their utility in the various cases of four different DCN topologies. We show that virtual bridging brings a high performance gain when TE is the primary goal and should be deactivated when EE becomes important. Moreover, we show that multipath forwarding can bring relevant gains only when EE is the primary goal and virtual bridging is not enabled.**

***Index Terms*—Virtual Bridging, Multipath Forwarding, Data Center Networking, VM Placement, Traffic Engineering.**

I. Introduction

The advent of efficient software virtualization techniques allows running server virtualization at competitive performance-cost trade-offs with respect to legacy solutions. The increasing adoption of server virtualization has recently favored three key technology advances in data-center networking: the emergence of virtual bridging functions at the hypervisor software level (between virtual machines and the physical network); a more efficient exploitation of path diversity by means of multipath forwarding protocols; the possibility to dynamically migrate virtual machines across virtualization servers at different places in the data-center network (DCN).

In the context of DCN optimization, virtual bridging is useful for the management of Virtual Machines (collocated in the same virtualization server), by offloading inter-Virtual Machine (VM) traffic from access and aggregation switches at the expense of an additional computing load on the virtualization server. Moreover, with the emergence of flat DCN topologies, such as Fat-Tree [3], DCell [4], and BCube [5] multipath forwarding can become useful to fully utilize the available paths and capacity and therefore offer higher throughput and resiliency to the servers. The ability to synchronize VM copies and migrate across virtualization servers (referred in the following also as 'containers' or 'VM containers') further adds elasticity to the cloud fabric by allowing fault-restoration and resource consolidation.

Virtual machine placement algorithms typically address traffic engineering (TE) [6] [7] or energy efficiency (EE) [8] [9] objectives, such as to minimize the maximum link utilization when balancing the traffic load on DCN links or to maximize server utilization to turn off or hibernate some servers to save energy. Addressing TE and EE goals eventually leads to savings in DCN maintenance and planning costs while increasing the performance. The relationship between the presented three recent trends, virtual bridging[1], multipath forwarding[2], and VM placement, is a rather unexplored subject that we investigate in this paper.

The contribution of the paper is two-fold:

- Given that, to the best of our knowledge, no work at the state of the art offers a DCN optimization framework supporting virtual bridging and multipath forwarding, we formally formulate the virtual machine placement optimization problem with these features in a novel, compact, and versatile formulation. For its resolution with dense, flat, and large DCN topologies, we propose a repeated matching heuristic.
- We analyze the impact of virtual bridging and multipath forwarding in DCN optimization with TE and EE objectives, and with a detailed sensibility analysis including four different DCN topologies (3-layer, FatTree, BCube, DCell) that cover all possible cases. We draw observations on the case-by-case suitability of virtual bridging and multipath forwarding features with respect to DCN VM placement optimization.

Section II presents the background of our work. The DCN optimization model is formulated in Section III, the proposed heuristic in Section IV, and simulation results are in Section V. Section VI concludes the paper.

D. Belabed, S. Secci, and G. Pujolle are with Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France (e-mails: dallal.belabed@upmc.fr, stefano.secci@upmc.fr, guy.pujolle@upmc.fr).

D. Medhi is with the University of Missouri-Kansas City, 5100 Rockhill Road Kansas City, MO 64110-2499, USA (e-mail: dmedhi@umkc.edu).

A preliminary version of the paper appears in the proc. of ITC 2014 [1], and some preliminary results also in the proc. of DCPERF'14 workshop [2].

This work was partially supported by the Systematic FUI 15 project RAVIR, the ANR Reflexion project (contract nb: ANR-14-CE28-0019), and the National Science Foundation grant CNS-0916505.

[1]The term 'virtual bridging' is used in the following to refer to the traffic switching operation at the software hypervisor level of the virtualization servers (VM containers). With virtual bridging enabled, the virtual bridge switches traffic between VMs in the same container, as well as traffic coming from outside and going outside the container.

[2]The term 'multipath forwarding' is used in the following to apply to the case when the traffic load at a given switching node can be balanced over multiple network paths.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2413755, IEEE Transactions on Network and Science Management

II. Background

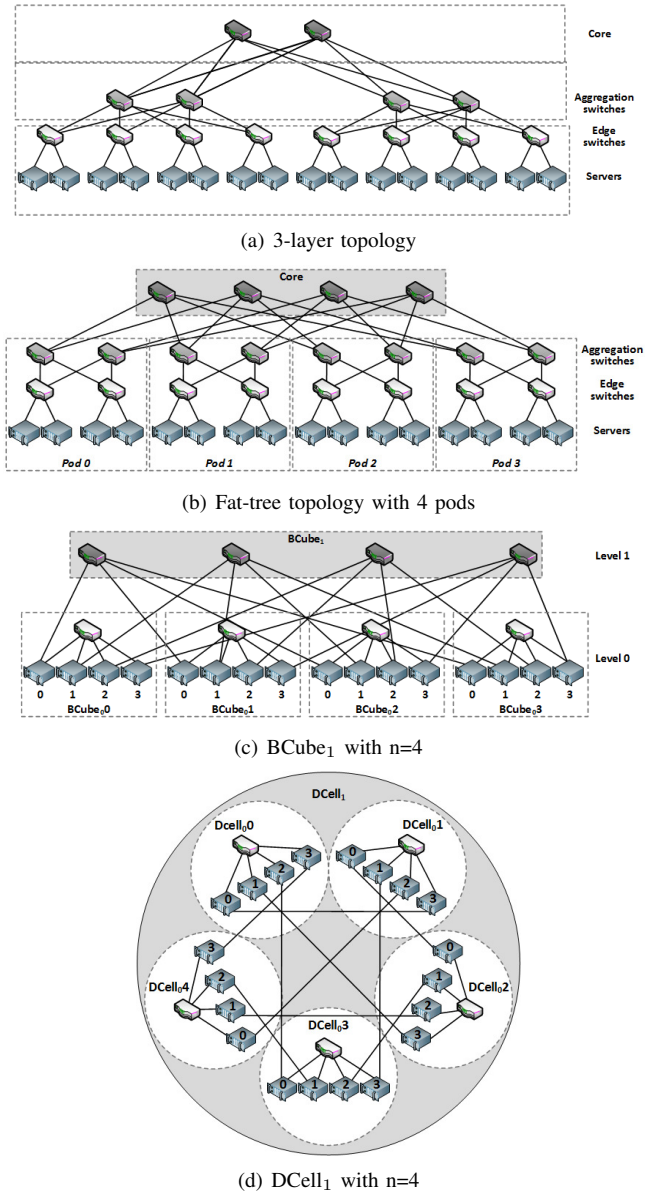
In order to have a clear understanding of the DCN optimization model and features, the following provides a detailed and selected state of the art work on DCN topologies, Ethernet routing, and consolidation formulations.

A. Data Center Network Topologies

The *3-layer* architecture [10], depicted in Fig. 1a, is the common legacy DC topology. Its three layers are at the access, aggregation, and core points. At the access layer, servers or server units (e.g., blades) are attached to the network, via access switches; at the aggregation layer, access switches connect to aggregation switches; at the core layer, each aggregation switch is connected to multiple core switches. Such an architecture typically relies on legacy Virtual Local Area Network (VLAN) and spanning-tree switching [11], which, while simple and fast, is known to underutilize the network resources. Even if TE mechanisms, such as a multiple spanning tree protocol (MSTP), root bridge priority and port cost optimization methods exist, major problems still persist, namely in terms of convergence time upon failures, routing, and physical topology changes.

As briefly mentioned in the introduction, alternative topologies have been proposed in recent years to better meet rising requirements of network virtualization and novel switching protocols. At the origin of the various propositions, the authors in [3] propose a special instance of a Clos topology called "Fat-Tree" to interconnect commodity-of-the-shelf (COTS) switches as a k-ary Fat-Tree. As in Fig. 1b, all switches are identical and are organized on two layers: the core layer and the pod layer. Generally, at the pod layer there are k pods, each one containing two layers of $\frac{k}{2}$ switches: edge switches and aggregation switches. Each k-port switch in the lower layer (edge layer) is directly connected to $\frac{k}{2}$ hosts. Each of the remaining $\frac{k}{2}$ ports is connected to $\frac{k}{2}$ of the k ports in the aggregation layer. Concerning the core layer, there are $(\frac{k}{2})^2$ k-port core switches. Each core switch has one port connected to each of the k pods. The i^{th} port of any core switch is connected to the i^{th} pod so that consecutive ports in the aggregation layer of each pod switch are connected to the core switches on $(\frac{k}{2})$ strides. Fig. 1b shows an example for $k = 4$.

Another proposed topology that captures major attention is BCube [5], a recursive modular architecture. As depicted in Fig. 1c, BCube has server devices with multiple ports (typically no more than four). Multiple layers of cheap COTS switches are used to connect those servers. A BCube_0 is composed of n servers connected to an n-port switch. A BCube_1 is constructed from n BCube_0s and n n-port switches. More generally, a BCube_k ($k \geq 1$) is constructed from n BCube_{k-1}s and n^k n-port switches. For example, in a BCube_k with n n-port switches, there are $k+1$ levels of switches. Each server has $k+1$ ports numbered from level-0 to level-k. Hence, BCube_k has $N = n^{k+1}$ servers. Each level has n^k n-port switches. The construction of a BCube_k is as follows. One numbers the n BCube_{k-1}s from 0 to $n-1$ and the servers in each BCube_{k-1} from 0 to n^k-1. Then one connects the level-k port of the i^{th} server ($i \in [0, n^k-1]$) in the j^{th} BCube_{k-1} ($j \in [0, n-1]$) to the j^{th} port of the i^{th} level-k switch. It is worth noting that BCube requires virtual bridging in the containers to operate. Fig. 1c shows an example of a BCube_1, with $n = 4$.



(a) 3-layer topology

(b) Fat-tree topology with 4 pods

(c) BCube_1 with n=4

(d) DCell_1 with n=4

Fig. 1: Data Center Network topologies.

Similarly to BCube, DCell [4] has servers equipped with many interfaces and COTS switches. A DCell server is connected to several other servers and a switch. A high-level DCell is constructed from low-level DCells. The connection between DCells can make use of virtual bridging. A DCell_k ($k \geq 0$) is used to denote a level-k DCell. DCell_0 is the building block to construct larger DCells. It has n servers and a switch ($n = 4$ for DCell_0 in Fig. 1d). All servers in DCell_0 are connected to the switch. In DCell_1, each DCell_0 is connected to all the other DCell_0s with one link; the Fig. 1d shows a DCell_1 example. DCell_1 has $n+1 = 5$ DCell_0s. DCell connects the 5 DCell_0s as follows. It assigns each server a 2-tuple $[a_1, a_0]$, where a_1 and a_0 are the level-1 and level-0 IDs, respectively. Thus a_1 and a_0 take values from $[0, 5)$ and $[0, 4)$,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2413755, IEEE Transactions on Network and Science Management

respectively. Then two servers with 2-tuples $[i, j-1]$ and $[j, i]$ are connected with a link for every i and every $j > i$. Each server has two links in DCell_1. One connects to its switch, and the other to a server in another DCell_0. In DCell_1, each DCell_0 is fully connected with every other virtual node to form a complete graph. Moreover, since each DCell_0 has n inter-DCell_0 links, a DCell_1 can only have $n+1$ DCell_0s as illustrated in Fig. 1d. A DCell_k is constructed in the same way to the above DCell_1 construction in a recursive procedure [4] that is more complex than the BCube one.

B. Ethernet fabric evolution

In the last decade, several evolutions of the legacy Ethernet switching architecture in terms of TE features have occurred. Under the perspective of incremental upgrades of the Ethernet switching architecture to meet TE requirements, one can consider MSTP [12] as the first attempt to actively perform TE in a legacy Ethernet switched network running STP, by default suffering from unused links due to STP port blocking. The multiplexing of multiple clients or VLANs into one among several spanning trees can also be optimized as presented in [12].

Other protocols trying to solve bottleneck issues along the spanning tree(s) have been standardized, notably the Link Aggregation Group and its evolutions [13] allows a switch to use multiple links as a single one with respect to STP. Eventually, the real TE bottleneck in Ethernet switching being the spanning tree fabric, the STP has been removed from recent carrier Ethernet solutions implementable in DCNs, namely: the Provider Backbone Bridges with Traffic Engineering (PBB-TE) [14] where centralized control servers push MAC tables to backbone switches (with a similar philosophy to OpenFlow [15]), the Layer-2 label switching architecture [16] using the VLAN fields as label-switching fields; the Shortest Path Bridging (SPB [17]) and the Transparent Interconnection of a Lot of Links (TRILL [18]) protocols where the control-plane is distributed adapting a layer-3 link state routing protocol (the Intermediate System to Intermediate System, ISIS, protocol). While SPB needs the whole DCN backbone to be upgraded with new switches, TRILL can be implemented only at key points (e.g., at the hypervisor level only using software implementations) and is therefore considered more scalable. As nodes in this context are no longer simple bridges since they perform a routing function in TRILL as well as in this paper, the term Router-Bridges (RBs) is adopted. While differing in terms of scalability and deployability, the latter three solutions have proven to be viable ones and have been adopted by many vendors. Notably, SPB and TRILL enabled multipath routing of Ethernet frames and hence, opened the way to active load-balancing over multiple paths across virtual and physical switches. In this paper, we assume DCN multipath capabilities are enabled by a protocol like these. It is worth mentioning that among these protocols, PBB-TE and OpenFlow can readily implement the result of an optimization framework such as the one described in this paper. On the other hand, as of now, SPB and TRILL are less versatile; apart from multipath load balancing, they do not offer or provide an easy way to support explicit TE routing.

C. Virtual Machine Placement in Data Center Networks

Relevant works propose VM placement algorithms that take network-specific constraints into consideration. In [19], the authors propose a VM placement algorithm considering network resource consumption, modeling a VM container as a set of CPU-memory slots where each slot can be allocated to a VM. They considered the number of VMs equal to the number of slots. The placement objective was to minimize the average traffic forwarding latency by assuming static single-path routing and focusing on two traffic models: a dense one, where each VM sends traffic to every other VM, and a sparse Infrastructure as a Service (IaaS)-like one with isolated clusters where only the VMs in the same IaaS communicate.

In [20], the authors present a VM placement algorithm based on a stochastic bin-packing modeling with non-deterministic demands following normal distributions and an ad-hoc resolution heuristic. In [21], the authors propose an iterative 3-step virtual embedding heuristic. During the first step, an arbitrary VM mapping is done and then the servers are mapped to the physical bridges. Finally, the virtual links are mapped accordingly. If one of these steps fail, the heuristic comes back to the previous step until a solution is found. However, the quality of the solution appears strongly dependent on the first step, and the back-tracking can lead to high time complexity.

In [6], the authors formulate the problem as a job placement problem, each job being associated with a number of VMs, by minimizing a global network cost. However, link capacity constraints as well as multipath forwarding capabilities are not considered. In [22], the authors formulate the problem as a routing problem, in order to address the network and server optimization problem as a single one. The objective of the optimization is the minimization of the energy consumption of the servers, bridges, and links and therefore discarding network-specific and server-specific cost components.

Some of these studies ignore link capacity constraints while others exclude dynamic routing as in [19], or consider the traffic volume to reduce the number of containers as in [20], or just the network resources as in [19] and [23]; only [22] considered multipath forwarding capabilities (in the sense that multiple egress paths are simultaneously used for the same destination). Commonly, because of the relatively recent employment of virtual bridging for transiting traffic at the server level, virtual bridging capabilities for external traffic forwarding are ignored. To the best of our knowledge, our study is the first one that precisely addresses the joint usage of virtual bridging and multipath forwarding, proposing adequate algorithms and assessing their impact on VM placement.

III. Optimization Problem

In the following, we present the mathematical formulation of the target VM placement problem. The optimization problem is to determine how to place VMs at VM containers in a DCN supporting virtual bridging and/or multipath forwarding while satisfying TE and/or EE goals. We first present the formulation with no multipath forwarding and virtual bridging capabilities, and then we show how it can be easily extended to enable these features. The notations are provided in Table I.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2413755, IEEE Transactions on Network and Science Management

TABLE I: Mathematical notations

N	set of VM containers and RBs; $n \in N$.
C	container set; $C \subset N$.
V	VM set; $V \subset N$.
R	RB set; $R \subset N$. $R_a \subset R$ is the access RB set.
T^V	set of VM pairs; $T^V \subset V \times V$.
T^C	set of container pairs; $T^C \subset C \times C$.
T^R	set of RB pairs; $T^R \subset R \times R$.
Variables and Parameters	
$e_{v,c}$	1 when v is at c, 0 otherwise. $v \in V, c \in C$.
b_c	1 if c is enabled, 0 otherwise. $c \in C$.
$a_{c,r}$	1 when c traffic via r. Multipath: $\in [0,1]$. $c \in C, r \in R$.
$\rho^k_{s,d}$	1 if traffic from r_s to r_d transits by the k^{th} path if unipath. Multipath: $\in [0,1]$. $(r_s, r_d) \in T^R$.
t_{c_i,c_j}	traffic from c_i to c_j, $(c_i, c_j) \in T^C$.
t_{r_i,r_j}	traffic from r_i to r_j; $(r_i, r_j) \in T^R$.
$t_{c,r}$	traffic from $c \in C$ to $r \in R$.
U	maximum network link utilization.
K^P_c	power capacity of container $c \in C$.
K^M_c	memory capacity of container $c \in C$.
d^P_v	computing power demand of VM $v \in V$.
d^M_v	memory demand of VM $v \in V$.
t_{v_i,v_j}	traffic from v_i to v_j, $(v_i, v_j) \in T^V$; $t_{v_i,v_i} = 0$.
$K_{i,j}$	(i,j) link capacity, null if no link; $(i,j) \in N \times N$.
$p^{k,s,d}_{i,j}$	1 when k^{th} path from r_s to r_d uses link (r_i, r_j).
α	trade-off coefficient between TE and EE objective, $\alpha \in [0,1]$.

We use a bi-criteria objective function that consists of the minimization of U, the maximum link utilization (TE goal), and the number of enabled containers (EE goal), weighted by the α trade-off factor instrumental to assess their impact on VM placement and DCN performance.

The objective is to balance between TE and EE goals:

$$\min \ \alpha \ U + (1-\alpha) \sum_{c \in C} b_c \tag{1}$$

while ensuring that a VM is assigned to only one container:

$$\sum_{c \in C} e_{v,c} = 1 \quad \forall v \in V. \tag{2}$$

A container is enabled only if it hosts at least one VM:

$$b_c \geq e_{v,c} \quad \forall c \in C, \ \forall v \in V. \tag{3}$$

Each container is assigned to one RB:

$$\sum_{r \in R} a_{c,r} = 1 \quad \forall \ c \in C. \tag{4}$$

Traffic between two access RBs is sent over a single path:

$$\sum_{k} q^k_{r_s,r_d} = 1 \quad \forall (r_s, r_d) \in R_a \times R_a. \tag{5}$$

A VM is assigned to a container only if there are available residual computing resources:

$$\sum_{v \in V} d^P_v \ e_{v,c} \leq K^P_c \quad \sum_{v \in V} d^M_v \ e_{v,c} \leq K^M_c; \quad \forall c \in C$$

$$\sum_{v \in V} d^P_v \ e_{v,c} \leq K^P_c \quad \sum_{v \in V} d^M_v \ e_{v,c} \leq K^M_c; \quad \forall c \in C. \tag{6}$$

Container-RB traffic does not violate the access link capacity:

$$t_{c,r} \leq K_{c,r} \quad \forall \ c \in C \quad \forall r \in R. \tag{7}$$

Similarly for the aggregation-core link capacity:

$$\sum_{r_s,r_d} \sum_{k} t_{r_s,r_d} \rho^k_{r_s,r_d} p^{k,r_s,r_d}_{r_i,r_j} < U \ K_{r_i,r_j} \quad \forall (r_i, r_j) \in T^R \tag{8}$$

$$t_{c,r} = \sum_{(v_i,v_j) \in T^V} (t_{v_i,v_j} + t_{v_j,v_i}) \ e_{v_i,c} \ a_{c,r} \quad \forall r \in R, \ \forall c \in C$$

$$t_{r_s,r_d} = \sum_{(c_i,c_j) \in T^C}^{i \neq j} t_{c_i,c_j} \ a_{c_i,r_s} \ a_{c_j,r_d} \quad \forall (r_s, r_d) \in T^R$$

$$t_{c_i,c_j} = \sum_{(v_x,v_y) \in T^V} (t_{v_x,v_y} + t_{v_y,v_x}) \ e_{v_x,c_i} \ e_{v_y,c_j} \quad \forall \ c_i, c_j \in C.$$

1) Enabling multipath capabilities: Multipath forwarding between containers and RBs (in the place of link aggregation/bonding or similar approaches) can simply be enabled by declaring $\alpha_{c,r}$ as a non-negative real variable. Hence, we add the following integrity constraint:

$$\sum_{k} a_{c,r} = 1 \quad \forall \ c \in C. \tag{9}$$

Similarly, multiple paths between RBs can be enabled by declaring $\rho^k_{s,d}$ as a non-negative real variable and adding the following integrity constraint:

$$\sum_{k} \rho^k_{r_s,r_d} = 1 \quad \forall (r_s, r_d) \in R_a \times R_a, \tag{10}$$

where the traffic between RBs in (8) becomes

$$t_{r_s,r_d} = \sum_{(c_i,c_j) \in T^C}^{i \neq j} t_{c_i,c_j} a_{c_i,r_s} \ a_{c_j,r_d}; \ \forall (r_s, r_d) \in T^R \ \forall \ c_i, c_j \in C.$$

2) Enabling virtual bridging: Enabling virtual bridging means that the container takes the function of a bridge (typically at the hypervisor level). This feature can be easily included by transforming the variable $a_{c,r}$ in a parameter and extending the RB set including the container nodes. Given that virtual bridging consumes additional power and memory, (6) should be slightly changed so that such an additional component (function of the traffic load) is included.

The use of virtual bridging consumes VM container power and memory. The constraints (6) change to (11) and the traffic between VMs in the same container no longer transits by physical bridges.

$$\tau(t_{c,c} + t_{c,r} + t_{r,c}) + \sum_{v \in V} d^P_v \ e_{v,c} \leq K^P_c; \quad \forall \ c \in C$$

$$\gamma(t_{c,c} + t_{c,r} + t_{r,c}) + \sum_{v \in V} d^M_v \ e_{v,c} \leq K^M_c; \quad \forall \ c \in C. \tag{11}$$

This optimization model is an extension of the baseline multi-commodity flow (MCF) problem for network routing with link capacity constraints [24] by taking into account peculiar data center networking constraints due to VM mobility, VM container switching on and off, virtual bridging, and multipath forwarding.

Given the elasticity related to VM migrations and multipath forwarding that require double mapping, between VMs and VM containers and between VM containers and usable paths, our optimization problem, even if comprehensive and versatile (considering both unipath and multipath modes with and without virtual bridging) is a non-linear one.

To summarize the problem is

$$\min \ \ (1) \ \ subject \ to: (2), (3), (4), (5), (6), (7) \ and \ (8)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2413755, IEEE Transactions on Network and Science Management

where (6) is replaced by (11) for the virtual bridging case. Contraints (9) and (10) are added for the multipath case. The problem has $|V| + 2|C| + |R_a|^2$ variables, $|V| + |C| + |R_a|^2$ with multipath forwarding and virtual bridging, and less than $|V| + |C|(2 + |V||R|) + 2|R|^2$ constraints.

IV. Heuristic approach

Classically, mapping problems reduce to facility location problems; when capacity constraints need to be verified as a function of the type of mapping, there are similarities with the capacitated facility location problem [25], in particular, to the single source facility location problem (SSFLP) [26], [27].

It is easy to derive that our DCN optimization problem can be reduced to the SSFLP, and hence, is NP-hard. In the SSFLP, we have a set of customers that must be served by a single facility, and there is a cost associated with opening a facility in a particular location and a transportation cost from the facility to the customer. Each customer has a particular demand and each facility has a limited capacity. The problem is to find where to locate the facilities to minimize the cost of the network. Our optimization problem can be reduced to an instance of a SSFLP as follows. Each VM pair can be seen as a customer of the SSFLP, and the VM pair has a global traffic demand that can be seen as the customer demand from a facility of the SSFLP. To translate our problem to a SSFLP instance, let (i) the traffic demand between two VMs of our VM placement problem correspond to a customer demand from a facility in the SSFLP; (ii) the cost of the link between the potential containers (where the two VMs can be located) and each access RBs correspond to half the cost between the potential facility location and the customer of the SSFLP; (iii) the first assignment of a VM to a container corresponds to the cost of opening a facility of the SSFLP; (iv) the container capacity constraint corresponds to the capacity of the facility in the SSFLP. In this way, the solution of such an SSFLP instance provides us with the solution of our VM placement problem and hence, NP-hard.

A. Reformulation of the optimization problem

Recently, modeling an optical network design problem as a facility location problem, the authors in [28] extended a repeated matching heuristic described in [26], [27] to solve the SSFLP and proved it can reach good optimality gaps for many large instances of the problem.

Motivated by those results and basic similarities with the problem in [28], we redesign our DCN optimization problem as a repeated matching problem. With respect to the network context of [28], we have more matching sets with peculiar constraints due to fundamental differences between optical networks and DCNs. The double mapping we have to handle in our problem and the multiple capacity constraints we have to care about (at both the link and VM container sides) makes this problem more combinatorial than [28], so that comparison to the optimum is not differently possible than in previous applications [26]–[28]. In our DCN scope, communications are between VMs that can be hosted behind the same VM container or behind distant containers interconnected by a DCN path. External communications can be modeled introducing fictitious VMs and VM containers acting as an egress point, if needed. When multipath is enabled, multiple paths can be used toward a same destination, and when virtual bridging is enabled, a VM container can transit external traffic if the topology supports it. When communicating VMs are not collocated, inter-VM communication should involve a pair of containers and at least a DCN path between them.

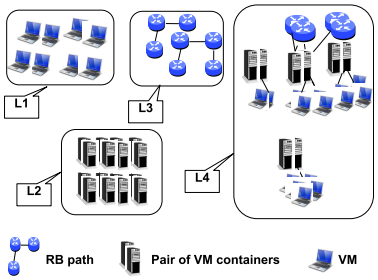


Fig. 2: Representation of heuristic sets: L_1, L_2, L_3, and L_4.

Let a VM container node *pair* be designated by cp, $cp \in T^C$, so that $cp = (c^i, c^j)$, i.e., a container pair is composed of two containers c^i and c^j). When $c^i = c^j$ the container pair cp is said to be *recursive*. A subset of container node pairs is designated by D^C so, $D^C \subseteq T^C$. Let the k^{th} path from RB r^1 to RB r^2 be designated by $rp = (r^1, r^2, k)$. A set of RB paths is designated by D^R so that $D^R \subset T^R$.

Definition IV.1. A **Kit** ϕ is composed of a subset of VMs D^V, a VM container pair $cp \in T^C$ and a subset of RB paths D^R. Each VM $v \in D^V$ is assigned to one of the containers in a pair cp (c^1, c^2). A container pair cp (c^1, c^2) is connected by each RB path rp $(r^1, r^2, k) \in D^R$, so that c^1 and c^2 are respectively mapped to r^1 and r^2. The Kit is recursive when its cp is recursive, and in such a case D^R must be empty. When the multipath is not enabled, $|D^R| = 1$. The Kit is denoted by $\phi(cp, D^V, D^R)$.

Definition IV.2. A Kit is a **Feasible Kit** if:

- D^V is not empty, i.e., $D^V \neq \oslash$.
- Memory and power demands of each VM are satisfied, i.e.(6), restricted to D^V and cp.
- In case of a non-recursive Kit, the link capacity constraints between VM containers are satisfied, i.e., (7) is restricted to D^V, D^R and cp.

Definition IV.3. $\mathbf{L_1}, \mathbf{L_2}, \mathbf{L_3}, \mathbf{and}\ \mathbf{L_4}$

- $\mathbf{L_1}$ is the set of VMs not matched with a container pair, i.e., $\mathbf{L_1} = \{v \mid v \in D^V \wedge v \notin \phi\}$.
- $\mathbf{L_2}$ is the set of VM container pairs not matched with an RB path, i.e., $\mathbf{L_2} = \{cp \mid cp \in T^C \wedge cp \notin \phi\}$.
- $\mathbf{L_3}$ is the set of RB paths not matched with a container pair, i.e., $\mathbf{L_3} = \{rp \mid rp \in T^R \wedge rp \notin \phi\}$.
- $\mathbf{L_4}$ is the set of Kits. It is worth mentioning that $\mathbf{L_4}$ becomes Packing when all its kits are feasible.

Definition IV.4. A **Packing Π** is a union of Kits in $\mathbf{L_4}$. A Packing is said to be feasible if it contains at least one feasible Kit and $\mathbf{L_1}$ is empty.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2413755, IEEE Transactions on Network and Science Management

B. Matching Problem

Given the DCN optimization problem elements using the above described sets, it can be reformulated as a matching problem. The classical matching problem can be described as follows. Let A be a set of q elements $h_1, h_2, \ldots, h_q$. A matching over A is such that each $h_i \in A$ can be matched with only one $h_j \in A$. An element can be matched with itself, which means that it remains unmatched. Let $s_{i,j}$ be the cost of matching h_i with h_j. We have $s_{i,j} = s_{j,i}$. Let $z_{i,j}$ be a binary variable equal to 1 if h_i is matched with h_j. The matching problem consists in finding the matching over A that minimizes the total cost of the matched pairs.

$$min \quad \sum_{i=1}^{q}\sum_{j=1}^{q} s_{i,j}\ z_{i,j} \tag{12}$$

$$s.t. \quad \sum_{j=1}^{q} z_{i,j} = 1, \quad i = 1, \ldots, q \tag{13}$$

$$\sum_{i=1}^{q} z_{i,j} = 1, \quad j = 1, \ldots, q \tag{14}$$

$$z_{i,j} = z_{j,i}, \quad i, j = 1, \ldots, q \tag{15}$$

$$z_{i,j} \in \{0, 1\}, \quad i, j = 1, \ldots, q. \tag{16}$$

(13) and (14) ensure that each element is exactly matched with another one. (15) ensures that if h_i is matched with h_j, then h_j is matched with h_i. (16) sets $z_{i,j}$ as binary.

In our heuristic, one matching problem is solved at each iteration between the elements of L_1, L_2, L_3, and L_4. At each iteration, the number of matchable elements is $n_1+n_2+n_3+n_4$ where n_1, n_2, n_3, and n_4 are the current cardinalities of the four sets, respectively. For each matching iteration, the costs $s_{i,j}$ have to be evaluated. The cost $s_{i,j}$ is the cost of the resulting element after having matched element h_i of L_1, L_2, L_3, or L_4 with element h_j. A basic example of matching is shown in Fig. 3: L_1 is empty ($n_1 = 0$), L_2 has two containers ($n_2 = 2$), L_3 has two RB paths ($n_3 = 2$), and L_4 has one feasible Kit ($n_4 = 1$). The result of the matching creates an unfeasible Kit, and modifies the existing feasible Kit with an additional RB path ($n_1 = n_2 = n_3 = 0$, $n_4 = 2$). At each iteration, the least-cost matching between the elements has to be determined. The computed matching costs $z_{i,j}$ are stored in a $(n_1 + n_2 + n_3 + n_4) \times (n_1 + n_2 + n_3 + n_4)$ cost matrix Z. Z dimensions change at each iteration, and Z is a symmetric matrix. Given the symmetry, only ten blocks have to be considered. The notation $[L_i - L_j]$ is used hereafter to indicate the matching between the elements of L_i and the elements of L_j as:

$$Z = \begin{pmatrix} [L_1 - L_1] & [-] & [-] & [-] \\ [L_2 - L_1] & [L_2 - L_2] & [-] & [-] \\ [L_3 - L_1] & [L_3 - L_2] & [L_3 - L_3] & [-] \\ [L_4 - L_1] & [L_4 - L_2] & [L_4 - L_3] & [L_4 - L_4] \end{pmatrix}$$

$$= \begin{pmatrix} [1] & [-] & [-] & [-] \\ [2] & [3] & [-] & [-] \\ [4] & [5] & [6] & [-] \\ [7] & [8] & [9] & [10] \end{pmatrix}.$$

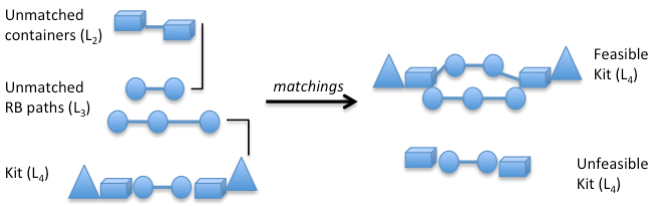


Fig. 3: A simple example of matching.

Selecting the least cost matching vector enables solution improvements via set transformations in the next iterations. Obviously, $L_1 - L_1$, $L_2 - L_2$ and $L_3 - L_3$ matchings are ineffective. To avoid a matching, e.g., because it is infeasible, its cost is set to infinity (a large number in practice). Matchings corresponding to other blocks without L_4 lead to the formation of Kits. Other matchings involving elements of L_4 shall lead to the improvement of the current Kits that also generate local improvements due to the selection of better VM containers or RB routes. Note that for such blocks, local exchange problems are to be solved for determining an exchange of VMs, VM containers and Kits, between the heuristic sets, while satisfying computing capacity constraints. The details on how to precisely compute each block matching costs are given in the Appendix.

The Kit cost computation has to maintain the same rationale as in the reference optimization problem when setting individual matching costs. The cost needs to be computed to de-motivate under-loading VM containers in terms of CPU and RAM utilization, while avoiding over-loading RB paths in terms of link utilization and respecting computing capacity constraints. The Kit cost function has to appropriately model two opposite forces due to the dual aspects stressing DCNs: computing and network resources. On the one hand, the Kit feasibility in terms of the link capacity constraints as described above, does not need to be enforced during the repeated matching iterations but rather to be motivated via the classical TE costs inducing the minimization of the maximum link utilization, and hence maximizing the minimum residual link capacity. On the other hand, residual computing capacities at the VM container level should be considered as costs. It is not suitable to have idle memory and CPU capacities when reducing the VM container is fixed energy consumption is one of the goals of the DCN provider. The overall Kit cost is not meant to represent a direct monetary cost, but it does. This is so that the repeated matching promotes more efficient Kits. Therefore, to align with the objective function (1), and by remembering that the cost of a Packing corresponds to the cost of its Kits, we set the cost of a Kit $\phi(cp, D^V, D^R)$ as:

$$\mu(\phi) = (1 - \alpha)\mu^E(\phi) + \alpha\mu^{TE}(\phi). \tag{17}$$

Where α is the trade-off scaling factor between the EE and the TE components, that are, respectively:

$$\mu^E(\phi) = \sum_{c^i \in cp} \left(\frac{K^P_{c^i}}{\sum_{v \in D^V_i} d^P_v} + \frac{K^M_{c^i}}{\sum_{v \in D^V_i} d^M_v} + \Gamma T_v \right) \tag{18}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2413755, IEEE Transactions on Network and Science Management

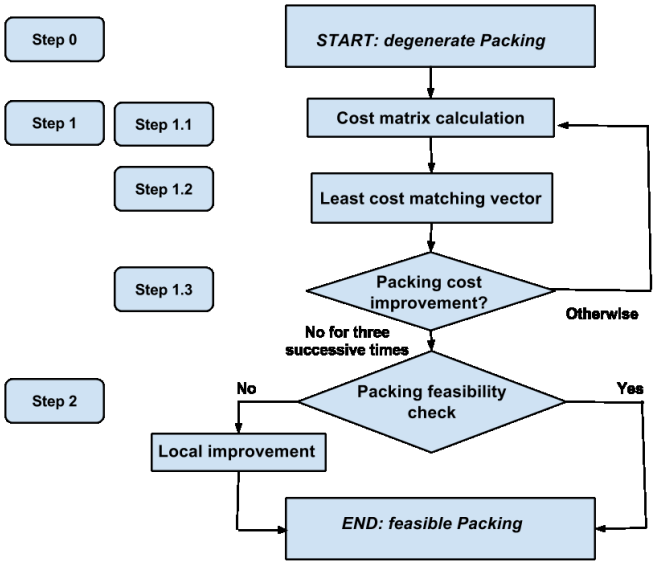


Fig. 4: Chart of the repeated matching heuristic steps.

$$\mu^{TE}(\phi) = \max_{(n_i,n_j)\in rp, rp\in\phi} U_{n_i,n_j}(\Pi). \quad (19)$$

Where T_v represents the global traffic v sends and receives, i.e., $T_v = \sum_{v'\in V}^{v\neq v'} t_{v,v'}$. Γ is the additional power and memory, to take into account the impact of traffic to the VM container CPU and memory consumption when virtual bridging is enabled (zero otherwise). Note that the computing capacity constraints are indirectly enforced within the $[L_4 - L_4]$ matching cost computation. $U_{n_i,n_j}(\Pi)$ is the link utilization of each link used by the current Packing Π solution, so that the maximum link utilization experienced by the Kit RB paths can be minimized. In our heuristic, in order to linearly compute the RB paths link utilization, the aggregation and core links of the RB paths are considered as congestion free, while the access container-RB links are considered as prone to congestion, which adheres to the reality of DCNs today. This is a realistic approximation we believe to be acceptable in a heuristic approach, especially because it allows a significant decrease of the time complexity.

C. Steps of the repeated matching heuristic

Due to the advantage of repeated matching between the different sets as described above, we can get rid of the non-linearities of the reference optimization problem with a heuristic approach that, based on the state of the art, is geared to achieve low optimality gaps. A global chart resuming the steps of our repeated matching heuristic is given in Fig. 4.

Its steps are as follows.

- **Step 0**: The algorithm starts with a degenerate Packing with no Kits and all other sets full.
- **Step 1**: A series of Packings is formed.
- **Step 1.1**: The cost matrix Z is calculated for every block.
- **Step 1.2**: The least cost matching vector is selected.
- **Step 1.3**: Go back to 1.1 for a new iteration unless the Packing cost has not changed in the last three iterations.
- **Step 2**: The heuristic stops, and in the case that L_1 is not empty, a local incremental solution is created assigning VMs in L_1 to enabled and available VM containers or, if none, to new containers.

TABLE II: Evaluated DCN setting cases.

VB	Multipath mode	Objective	Topologies
yes	MRB	EE	BCube, DCell
		TE	BCube, DCell
		EE+TE	BCube, DCell
no	MRB	EE+TE	3-layer, Fat-Tree, BCube*,DCell*
	MCRB	EE+TE	BCube**
	MRB-MCRB	EE+TE	BCube**

The least cost matching computation (Step 1.2) can be hard to optimally solve because of the symmetry constraint (15). In our heuristic, we decided to solve it in a suboptimal way to lower the time complexity. We have implemented the algorithm in [29], based on the method of Engquist [30]. Its starting point is the solution vector of the matching problem without the symmetry constraint (15) obtained with the algorithm described in [31] that was chosen for its speed performance. Its output is a symmetric solution matching vector.

For instance, if the resulting Packing in Fig. 3 does not change for three times, the result composed of the feasible kits (one Kit in the example) is kept as final result. Designing the matching costs in an efficient and rational way, the Packing cost across iterations should be decreasing - the decrease is expected to be monotonic starting by the moment when L_1 gets empty, so that the heuristic converges.

D. Time complexity

The complexity of the whole heuristic depends on its different sub-algorithms and phases. The calculation of the cost matrix is straightforward except for two blocks of the matrix (see blocks 10 and 8 in the Appendix) where a polynomial swapping problem depends on the number of connections in the network. The resolution of the matching problem operates on the cost matrix through the Forbes's and the Volgenant's algorithms. In the worst case, the first has a $\ominus(n^3)$ complexity while the second one has a $\ominus(n^2)$ complexity, where $n = n_1 + n_2 + n_3 + n_4$.

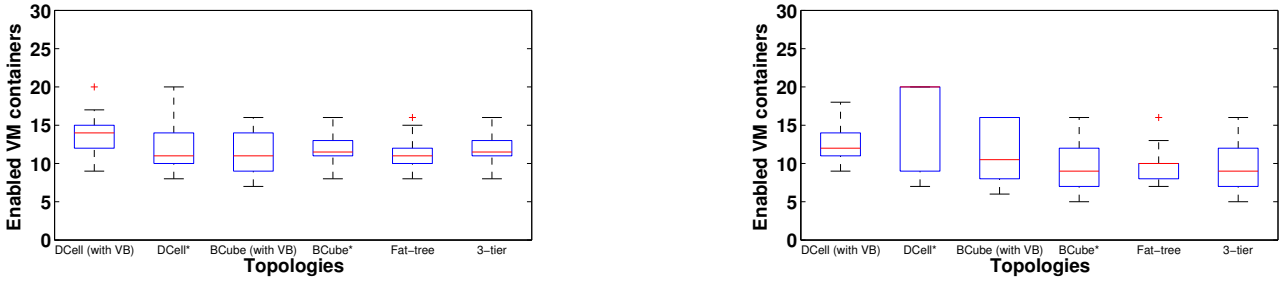
V. Simulation results

We implemented our heuristic using Matlab, and CPLEX for the computation of matching costs of some blocks. The adopted VM containers correspond to an Intel Xeon 5100 server with 2 cores of $2.33GHz$ and $20GB$ RAM and able to host 16 VMs. We use various weights for the TE and EE components in the optimization objective, and we analyze what happens when multipath forwarding and virtual bridging are enabled. We use the different forms of multipathing, encompassing the following cases.

1) Multipathing between RBs (MRB).
2) Multipathing between containers and RBs (MCRB).
3) Both multipathing modes (MRB-MCRB).

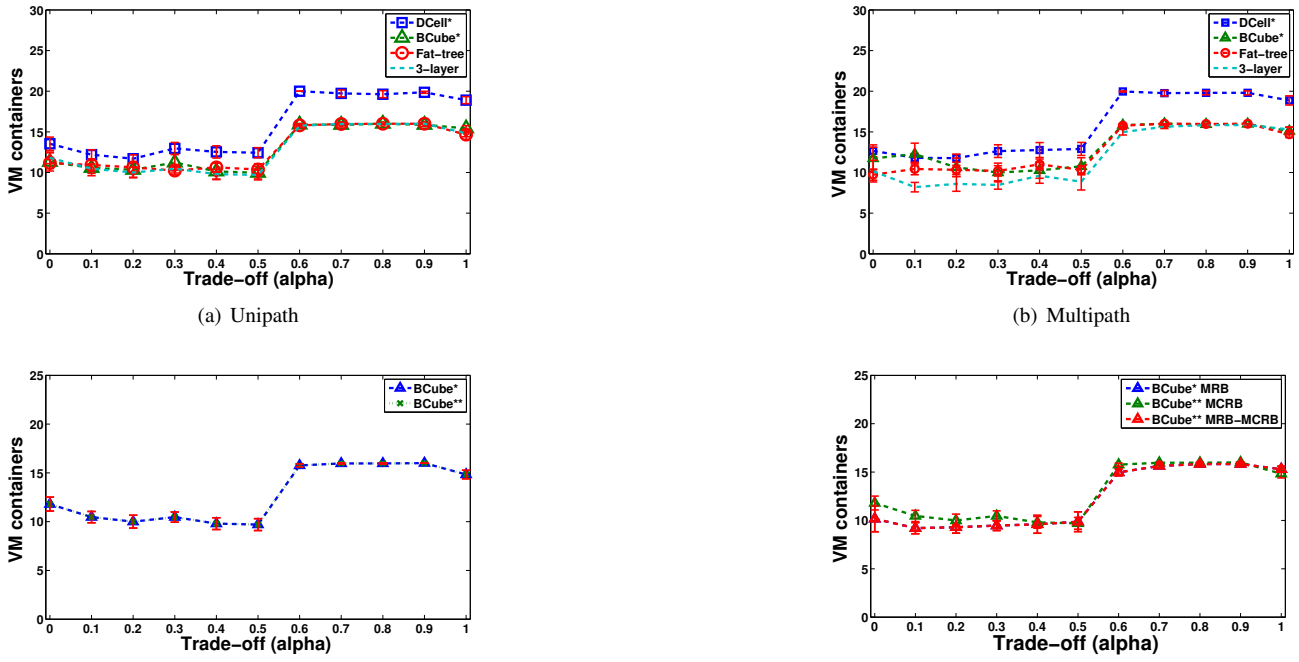
We executed our heuristic with the 3-laye Fig. 1a, Fat-Tree Fig. 1b, BCube Fig. 1c and DCell Fig. 1d DCN topologies. We note that BCube and DCell work properly only by employing virtual bridging at the server level. We allowed for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2413755, IEEE Transactions on Network and Science Management



(a) Unipath (b) Multipath

Fig. 5: Number of enabled VM containers Energy Efficiency (EE) results, with EE as single objective (VB=Virtual Bridging).



(a) Unipath (b) Multipath

(c) Unipath (d) Multipath

Fig. 6: Number of enabled VM containers EE results, without virtual bridging.

TABLE III: Evaluated DCN size cases.

Topologies	Container number	Container capacity	VM number
Fat-tree, BCube	16	16	150
DCell	20	16	240

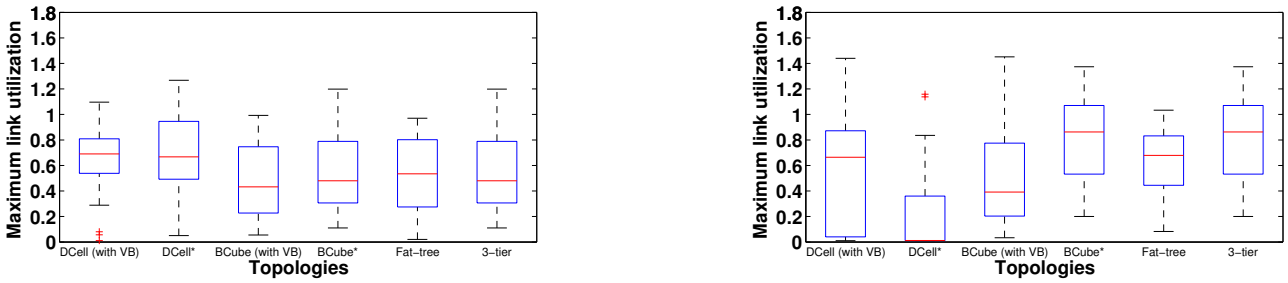
a small modification of the topology to allow a reference comparison between them and the other topologies for the cases without virtual bridiging, calling these variations BCube* and DCell*, respectively. We allow the access switches in DCell* to be directly connected to each-other, and we allow each access switch in BCube* to be directly connected to all core switches. Moreover, the 3-layer, Fat-Tree and DCell topologies have no multipath forwarding capabilities between containers and RBs, because there are no multiple links between containers and RBs (only BCube has that specificity). To also evaluate the case with multipath between RBs and containers, we use another variation to BCube, referred to as BCube**, where each server is multi-homed with two switches, its pod switch and one core switch. Table II summarizes the topologies that do or do not support the virtual bridging mode for each multipath forwarding case.

In the simulations, all DCNs are loaded at 70% in terms of computing capacity. Table III summarizes each topology's container capacity, the number of VMs that a container can host, and the total number of VMs used for our simulations. Note that with all topologies, we allowed for a certain level of overbooking in the resource allocation area for the sake of algorithm fluidity, especially at starting and intermediate iterations. The capacity of the access link was set to 10 $Gbps$.

We ran 30 different instances with different traffic matrices for each case. The reported results have a confidence interval of 95%. Our heuristic reached convergence roughly within a dozen minutes of each execution and successfully reaching steady states.

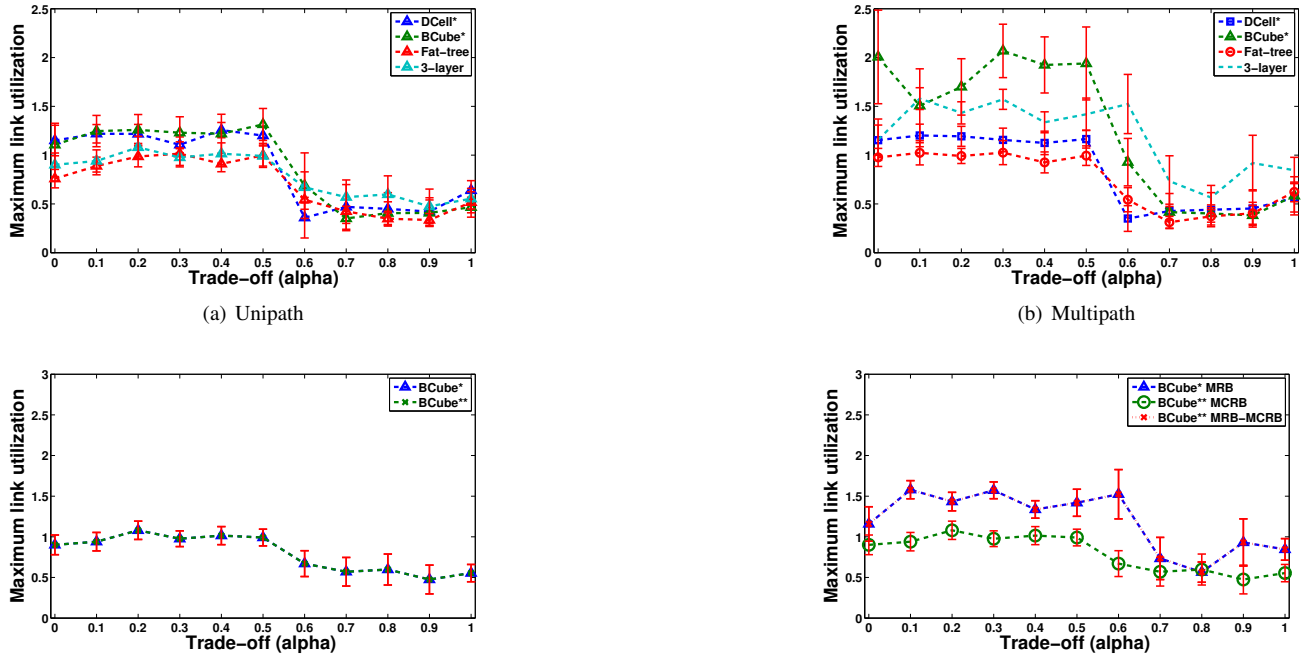
In the following, we detail the adopted traffic model and related state of the art. Then, we report experiment results we obtained considering the impact of virtual bridging and multipath forwarding on EE-oriented and TE-oriented VM placement. We look at the impact of virtual bridging and multipath forwarding without mixing the TE and EE objectives, and we perform sensibility analysis by varying the EE-TE α

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2413755, IEEE Transactions on Network and Science Management



(a) Unipath (b) Multipath

Fig. 7: Maximum link utilization Traffic Engineering (TE) results, with TE as single objective (VB=Virtual Bridging).



(a) Unipath (b) Multipath

(c) Unipath (d) Multipath

Fig. 8: Maximum link utilization TE results, without virtual bridging.

trade-off coefficient.

A. Traffic model

Choosing an appropriate traffic model is important when running DCN simulations. In the state of the art there are a few relevant works. We built our traffic drawing conclusions from the following studies [19], [32]–[34]. Authors in [32], [33], collected data from 3-layer-like DCNs and found that the traffic originating from a rack showed an ON/OFF behavior following heavy-tailed distributions. 80% of the DC server-originated traffic stayed within the rack, while between 40% and 90% left the rack. In [34], more than 90% of transfers had a volume from 100 MB to 1 GB. Moreover for 50% of its running time, a VM handled approximately 10 concurrent flows, and at least for 5% of its running time, a VM had more than 80 concurrent flows. In [19], the authors analyzed the incoming and outgoing traffic rates of an IaaS DCN with 17,000 VMs, reporting that 80% of the VMs had an average rate less than 800 KB/min. However, 4% of them had a ten times higher rate. Moreover, the traffic rate standard deviation was 82%, lower than or equal to twice the mean traffic.

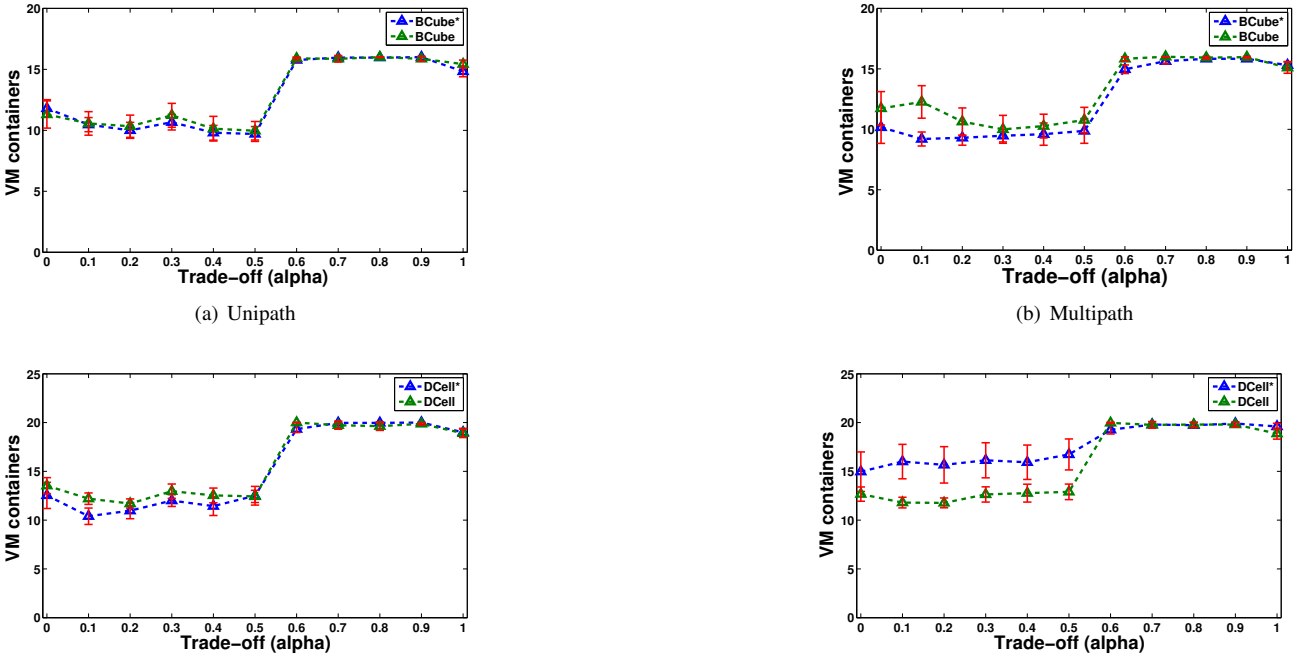
Since not all VMs communicate with each other in todays DCNs adopting network virtualization, but instead traffic is segmented by IaaS networks, we built an IaaS-like traffic matrix as in [19], which somehow also takes into account the IaaS traffic rack vicinity trend reported in [32], [33]. We apply the experimental incoming and outgoing traffic distribution of [19], with IaaS clusters of up to 30 VMs communicating with each-other and not communicating with other IaaS VMs. Within each IaaS, the traffic matrix was built according to the traffic distribution of [34].

B. Energy efficiency considerations

Fig. 5 illustrates the results in terms of enabled VM containers for different topologies when EE is the unique goal (i.e., $\alpha = 0$ in the problem formulations). We report results for both the cases when multipath forwarding is not enabled (i.e., $|D^R| = 1$ for all Kits) and the case where it is enabled. By observing the results we can assess the following:

- *Impact of virtual bridging*: When the EE is the goal, virtual bridging leads to negligible differences in EE performance;

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2413755, IEEE Transactions on Network and Science Management



(a) Unipath (b) Multipath

(c) Unipath (d) Multipath

Fig. 9: Number of enabled VM containers EE results, with virtual-bridging.

- *Impact of multipath forwarding*: It has a positive impact on EE when virtual bridging is disabled, and seems to be counterproductive when virtual bridging is enabled;
- *Sensibility to topologies*: The DCell topology shows a better EE performance than the BCube, especially when multipath forwarding is enabled. This can be explained by the higher path diversity at the DCell container. Hierarchical topologies, Fat-Tree, and 3-layer, show an overall worst EE performance for single-path forwarding and a better EE performance for multipath forwarding, with a negligible difference.

This analysis suggests that enabling virtual bridging does not bring any useful EE gain and can even worsen the EE performance when the consolidation EE objective is minimizing the number of enabled VM containers. In the following, we discuss the results with an EE perspective when we vary the impact of the EE goal in the VM placement.

1) Optimization goal sensibility with respect to multi-path forwarding features: With respect to multipath forwarding features, Figs. 6 illustrates the results in terms of enabled VM containers for different values of the trade-off parameter α between the EE and TE goal, ranging from a null value giving full importance to the EE goal, to a maximum value giving importance to the TE goal, with a step of 0.25.

We report the results including the case when multipath is not enabled (i.e., $|D^R| = 1$ for all Kits) and the case where it is enabled. Observing the results we can assess that:

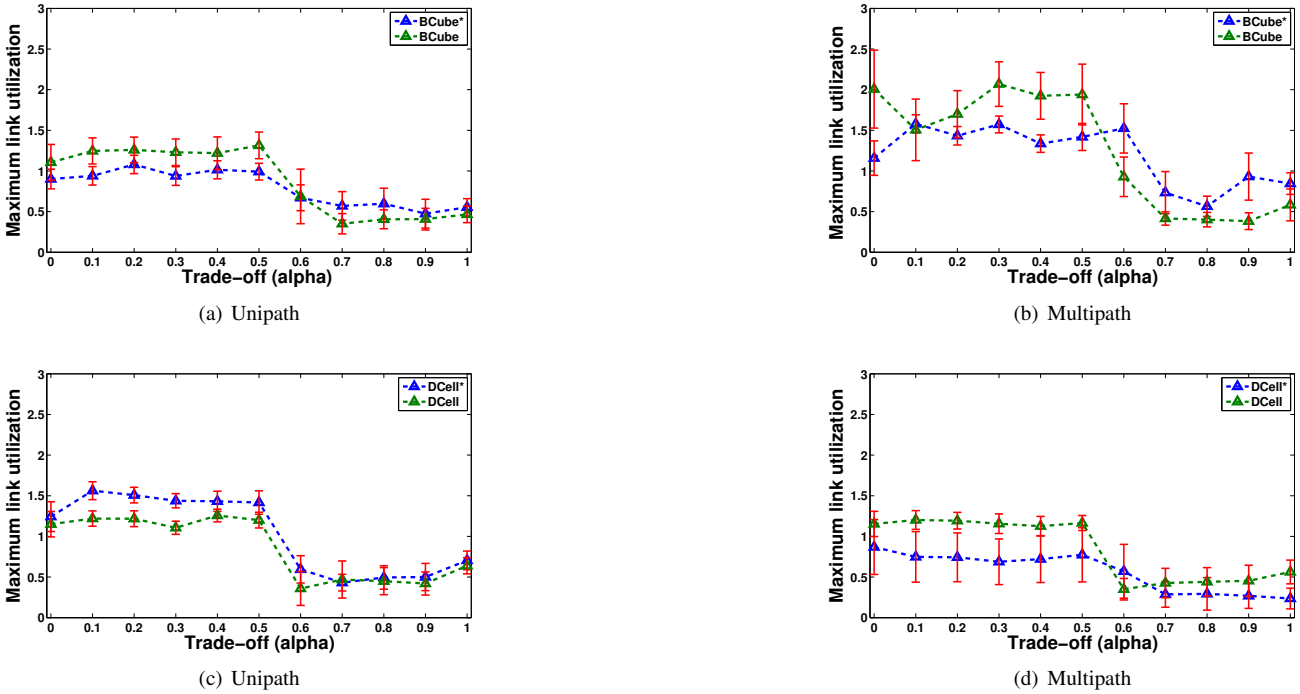
- When EE is discarded ($\alpha = 1$), the number of VM containers reaches its maximum, which could be expected.
- The results for all topologies are similar for MRB - the DCell slope is slightly higher then the other ones, which can be explained by the number of containers within DCell topology, equal to 20.
- For MRB, the enabling of multipath forwarding decreases roughly up to 30% the number of enabled VM containers, and only by 20% for MCRB when EE is considered as an important objective.
- The impact of multipath forwarding becomes negligible when EE is not considered important;
- MRB-MCRB gives the same effect as enabling MRB.

These results are counter-intuitive. On the one hand, decreasing the access link bottleneck by enabling multipath L2 forwarding seems to free VMs and allow a better VM consolidation switching off unused containers that lead to energy gains. On the other hand, multipath communications appear not to be useful for that goal when switching off VM containers is either not interesting or not possible.

2) Optimization goal sensibility with respect to virtual-bridging features: Furthermore, we analyze the results with an EE perspective when both virtual bridging and multipath forwarding features are enabled. Fig. 9 illustrates the results in terms of enabled VM containers for different values of the trade-off parameter α, ranging from a null value giving full importance to EE goal, to a maximum value giving importance to the TE goal. The figure reports the results for BCube and DCell topologies that support virtual bridging when multipath is enabled or not. Observing the results we can assess that:

- Enabling only virtual bridging (Fig. 9(a,c)) does not have a relevant impact on the number of enabled VM containers. We note a negligible gain when EE is the goal ($\alpha = 0$), for BCube, and the opposite for DCell topologies (both have the same confidence interval).
- Enabling multipath forwarding (Fig. 9(b,d)) has a positive EE impact when virtual bridging is disabled (the two curves have the same confidence interval when virtual bridging is enabled) - this impact becomes negligible

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2413755, IEEE Transactions on Network and Science Management



(a) Unipath (b) Multipath

(c) Unipath (d) Multipath

Fig. 10: Maximum link utilization TE results, with virtual bridging.

when EE is not considered as an important.

- We note that DCell topology shows better EE results than the BCube topology. In fact, the DCell topology used $\sim 55\%$ of the containers while BCube used $\sim 69\%$ of the containers.

These results confirm that, on the one hand, enabling virtual bridging has no impact on the EE goal regardless of the EE-TE trade-off level. On the other hand, multipath forwarding has a positive impact only when the virtual bridging is disabled.

C. Traffic engineering considerations

As already discussed, EE goals can be considered as opposite to the TE goals. Chasing EE tends to minimize the number of enabled VM containers, yet no care is given to network link utilization.

Fig. 7 reports the results when TE is the single goal of the virtual machine placement optimization, i.e., $\alpha = 1$, considering singlepath and multipath forwarding for the different topologies. We make the following observations:

- *Impact of virtual bridging*: Virtual bridging lead to sensible TE performance gains.
- *Impact of multipath forwarding*: With singlepath forwarding, the DCell case gets the largest TE gain, from a median of roughly 65% of the maximum link utilization to roughly 45% - this is due to the fact that virtual bridging in the DCell allows for indirectly minimizing the number of links used to interconnect servers (with multipath forwarding, the BCube case gets the largest TE gain, with the maximum link utilization being halved from about 80% to 40%);
- *Sensibility to topologies*: BCube and DCell do have similar TE performances, with a slighter better performance with the BCube probably because the gain in path diversity brought by virtual bridging is higher with the BCube (which keeps a core layer unlike the DCell) - the TE gain with respect to hierarchical topologies (Fat-tree, 3-layer) is always positive and slightly higher with enabled multipath forwarding.

These TE performance results are not intuitive and appear to be quite relevant. It is interesting to adopt virtual bridging when the primary goal of the virtual machine placement is traffic engineering. Flat topologies show a sensible gain with respect to more hierarchical topologies, which once more motivate the migration to such new topologies for IaaS-based DCNs.

1) Optimization goal sensibility with respect to multipath forwarding features: Under a TE perspective, the lower the maximum utilization is, the better it is, in order to ensure highly efficient communications. Under this perspective, we perform a sensibility analysis varying the TE-EE trade-off. Fig. 8 reports the maximum link utilization for both the unipath and the multipath cases, under different trade-off coefficients.

As expected, the curve decreases with α (oppositely to the previous curve, observing EE performance, in Fig. 6). We make the following observations:

- MRB can be counterproductive: the unipath case guarantees a better TE performance when TE is not considered as an important goal in DCN optimization (i.e., when $\alpha \rightarrow 0$) - the MRB mode induces unacceptable TE performance when TE is not the primary goal.
- The curves of all topologies are similar for the unipath case - DCell has the worst curve when EE is the goal, and all curves converge when the maximum importance

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2413755, IEEE Transactions on Network and Science Management

is given to TE.
- MRB-MCRB gives the same effect as enabling MRB.

2) Optimization goal sensibility with respect to virtual-bridging features: Furthermore, we analyse the impact of virtual bridging under a TE perspective. Fig. 10 illustrates the results in terms of maximum link utilization for different values of the trade-off parameter, considering the case when multipath is enabled and when it is not enabled, for the DCell and BCube topologies. We make the following observations:

- With respect to the BCube case (Fig. 10a-b), enabling virtual bridging when EE is the goal has a negative impact on the maximum link utilization and a positive one when TE is the goal.
- Accordingly to Fig. 10c, the DCell topology is less impacted by virtual bridging than BCube, which can be explained by the totally flat nature of DCell (with its single access layer).
- Enabling multipath forwarding appears to have a negative impact on TE when TE is not the goal, regardless of the topologies.

We note that virtual bridging has a positive impact when TE is the goal, but it can have a negative TE impact when EE is the goal, and the topology that benefites the most from virtual bridging is BCube.

VI. Summary

Data center networking is a challenging field of applications of old and new technologies and concepts. In this paper, we investigate how traffic engineering (TE) and energy efficiency (EE) goals in virtual machine placement can coexist with the emergence of virtual bridging (i.e., the capability to switch traffic at the hypervisor level in virtualization servers) and of multipath forwarding (i.e., the capability to balance the load toward a same destination on multiple egress paths). We provide a versatile formulation of the virtual machine placement problem supporting virtual bridging capabilities and multipath forwarding and propose a repeated matching heuristic for its resolution.

Through extensive simulation of realistic instances with legacy and novel flat DC topologies (i.e., 3-layer, Fat-Tree, BCube, and DCell), we found that:

- *Impact of multipath forwarding*: Multipath forwarding has a positive impact on EE when virtual bridging is disabled, and this positive impact becomes negligible when virtual bridging is enabled or EE is not considered as important.
  Enabling multipath forwarding between virtualization servers (containers) and router-bridges (RBs) does not bring relevant additional performance gains with respect to both EE and TE goals. This suggests that legacy link aggregation/bonding protocols between servers and bridges are sufficient and multipath routing protocols do not need to be brought down to the hypervisor level.
- *Impact of virtual bridging*: When TE is the primary goal, virtual bridging shows notable positive gains, and the TE performance gain can be important and can be improved up to two times, with a maximum link utilization that can be halved for the BCube DCN topology, while remaining important also for the DCell topology. Flat topologies show a notable gain with respect to more hierarchical topologies (3-layer, Fat-Tree) that once more motivate the migration to such new topologies for IaaS-based DCNs.
- *Sensibility to DCN optimization goals*: When EE is the primary goal of the DCN optimization, both multipath forwarding and virtual bridging can be counter-productive, leading to saturation at some access links. Instead, when TE is the primary goal, multipath forwarding grants only a moderate gain.

We believe these results provide important insights on the possible joint or individual adoption of multipath forwarding protocols and virtual bridging in data-center networks, to help data center network designers take the right choices when energy efficiency and traffic engineering performance goals need to be taken into account.

Appendix A
Matching cost matrix computation

Block 1: Matching two VMs. Given that the matching does not make real sense, its cost is set to infinity.

Block 2: Matching a container pair with a VM. This matching forms a Kit $\phi(cp, \overline{D}^V, \overline{D}^R)$ with $\overline{D}^R = \oslash$ and $\overline{D}^V = \{v\}$. Let cp be the i^{th} element of L_2 and v be the j^{th} element of L_1. With a recursive cp, the matching cost is

$$z_{n_1+i,j} = \sum_{s \in S} K^s_{c^1}/d^s_v. \tag{20}$$

Otherwise, the VM is assigned to the container with the least cost, i.e.:

$$z_{n_1+i,j} = \min\{\textstyle\sum_{s\in S} \frac{K^s_{c1}}{d^s_v}, \sum_{s\in S} \frac{K^s_{c2}}{d^s_v}\} + \Theta. \tag{21}$$

For this latter case, Θ is a big constant so that $\Theta >> \frac{K^s_{c^i}}{d^s_{v_j}}, \forall i, \forall j$; it is meant to discourage the matching over lower cost matchings, given that the created Kit is unfeasible (no route between containers), and it has an unused container.

Block 3: Matching two container pairs. Given that the matching does not make real sense, its cost is set to infinity.

Block 4: Matching an RBridge path with a VM. Given that the matching does not make real sense, its cost is set to infinity.

Block 5: Matching an RBridge path with container pair. Let rp be the i^{th} element of L_3 and cp be the j^{th} element of L_2.

$$z_{n_1+n_2+i,n_1+j} = \begin{cases} 2\Theta & \text{if } c^1 \neq c^2 \\ \infty & \text{otherwise.} \end{cases} \tag{22}$$

In the first case, an unfeasible Kit is formed with both containers unused, while the second case is impossible as it is pointless.

Block 6: Matching two RBridge paths. Given that the matching does not make real sense, its cost is set to infinity.

Block 10: Matching two Kits. Let $\phi_i(cp_a, D^V_a, D^R_a)$ be the i^{th} Kit of L_4 and $\phi_j(cp_b, D^V_b, D^R_b)$ be the j^{th} Kit of L_4. For $i = j$, since an element cannot be matched with itself, the matching cost is equal to the cost of the Kit; otherwise, several cases are possible:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2413755, IEEE Transactions on Network and Science Management

Case 1) All VMs are assigned to cp_a. The matching cost is therefore equal to the cost of the new Kit $\phi'(cp_a, D^V = D_a^V \cup D_b^V, D_a^R)$, i.e., $v_1 = \mu(\phi')$. Assignment of VMs to Kit container is the result of an optimization so that the cost of the Kit is minimized as follows.

$$v_1 = min\ (1-\lambda) \sum_{c}^{cp} \sum_{v}^{D^V} \left(\sum_{s \in S} \frac{K_c^s}{d_v^s} + \Gamma T_v^c \right) x_v^{c_i} + \lambda U \quad (23)$$

subject to

$$\sum_{c_i \in cp_a} x_v^{c_i} = 1, \quad \forall v \in D^V \quad (24)$$

$$\sum_{v_i \in D^V} \sum_{v_j \in D^V}^{i \neq j} t_{v_i}^{v_j} x_{v_i}^c + l_e \leq K_e U, \quad \forall e \in E_a \quad (25)$$

$$\sum_{v_i \in D^V} \sum_{v_j \in D^V}^{i \neq j} (t_{v_i}^{v_j} x_{v_i}^c + t_{v_i}^{v_j} x_{v_j}^c) = T_v^c, \quad \forall c \in cp_a. \quad (26)$$

where $x_v^{c_i}$ is a binary variable equal to 1 if $v \in D^V$ is assigned to $c_i \in cp_a$ (0 otherwise); $E = \{(c,r),(r,c) | c \in cp, r \in D^R\}$, is the set containing the two links from container c along the route r, and from r to c, and $E_a = \{(c,r),(r,c) | c \in cp_a, r \in D^R\} \subset E$; l_e is the load of the access link $e \in E$ as of last Packing configuration without ϕ_i and ϕ_j; K_e are the link capacities. Constraint (24) guarantees each VM is assigned to only one container; (25) computes U. The objective function also includes cost components inversely proportional to system resources utilization.

Case 2) All VMs are assigned to cp_b. Similarly to the previous case, the new kit $\phi''(cp_b, D_a^V \cup D_b^V, D_b^R)$ is formed with $v_2 = \mu(\phi'')$.

Case 3) The container pairs (cp_a, cp_b) and the set of RB paths (D_a^R, D_b^R) do not change, and VMs can be exchanged between Kits. That is, every $v \in D_a^V \cup D_b^V$ can swap its container.

We then need to find the optimal pair assignment. The cost of the new situation should be less then the previous one. This can be formulated as an integer linear problem. Let us define w_v as a binary variable so that $w_v = 1$ if the $v \in D_a^V$ swaps its current Kit ϕ' for ϕ'', and $w_v = 0$ otherwise; in order to avoid non linearities, we admit that v swaps its current $c^{i^{th}} \in cp_a$ for the $c^{i^{th}} \in cp_b$, i.e., the destination container is in the same indexed position in the container pair than in the source container pair, so $x_v^{c_i}$ is considered as a parameter and no longer as a variable in the following. Also, let z_v be a binary variable equal to 1 if $v \in D_b^V$ swaps its current Kit ϕ'' for ϕ', and 0 otherwise. Finally, g_v and h_v are the marginal costs formed by the pair exchanges (i.e., the difference between the new and the old cost for each $v \in D_a^V \cup D_b^V$), defined as

$$g_v = \mu(\phi'' \cup \{v\}) - \mu(\phi') \quad (27)$$

$$h_v = \mu(\phi' \cup \{v\}) - \mu(\phi''). \quad (28)$$

The swapping problem can be formulated so as to minimize the cost of the Packing:

$$v_3 = \min \sum_{v \in D_a^V} g_v\ w_v + \sum_{v \in D_b^V} h_v\ z_v. \quad (29)$$

The required constraints are omitted because of space limitation - they are needed to check the surplus power and memory capacity constraints for the containers, the surplus capacity constraints for the paths between the container nodes and each Rbridge node, as well as to set that the variables' domain.

Among the cases, we choose the least cost one:

$$z_{n_1+n_2+n_3+i, n_1+n_2+n_3+j} = min\{v_1, v_2, v_3\}. \quad (30)$$

Block 8: Matching a Kit with container pair. Let $\phi_1(cp_a, D_a^V, D_a^R)$ be the i^{th} Kit of L_4 and cp_b be the j^{th} pair of L_2. The cp_b can be seen as a Kit with no assigned $D_a^V = D_a^R = \oslash$: $\phi_2 = (cp_b, \oslash, \oslash) \in L_4$. The matching cost is then the cost of the new kit $\phi(cp_b, D_a^V, D_a^R)$:

$$z_{n_1+n_2+n_3+i, n_1+j} = \quad \mu(\phi). \quad (31)$$

Block 7: Matching a Kit with a virtual machine. Let $\phi_1(cp_a, D_a^V, D_a^R)$ be the i^{th} Kit of L_4 and $q(v^1, v^2)$ be the j^{th} pair of L_1. Three cases can be considered.

Case 1) The Kit is as $D_a^V = \oslash$: $\phi(cp_a, \oslash, D_a^R)$. Then D_a^V becomes $D_a^V = \{q\}$. The matching cost for this case is

$$z_{n_1+n_2+n_3+i,j} = \mu(\phi). \quad (32)$$

Case 2) The Kit has $D_a^R = \oslash$, i.e., $\phi(cp_a, D_a^V, \oslash)$. The virtual machine can be assigned to the container cp_a. Then D_a^V becomes $D_a^V \cup \{q\}$. The matching cost for this case is (now $q \in D_a^V$):

$$z_{n_1+n_2+n_3+i,j} = \begin{cases} \mu(\phi) & \text{if } c^1 = c^2 \\ \infty & \text{otherwise.} \end{cases} \quad (33)$$

Case 3) The Kit is as $\phi(cp_a, D_a^V, D_a^R)$. Then D_a^V becomes $D_a^V \cup \{q\}$. Noting that now now $q \in D_a^V$, the matching cost for this case is

$$z_{n_1+n_2+n_3+i,j} = \mu(\phi). \quad (34)$$

Block 9: Matching a Kit with RBridge path. Let $\phi(cp_a, D_a^V, D_a^R)$ be the i^{th} Kit of L_4 and $q(r^1, r^2)$ be the j^{th} pair of L_3. Three case can be considered:

Case 1): The Kit is as $D_a^V = \oslash$: $\phi(cp_a, \oslash, D_a^R)$. The matching for this case is impossible, i.e.:

$$z_{n_1+n_2+n_3+i, n_1+n_2+j} = \infty. \quad (35)$$

Case 2): The Kit is as $D_a^R = \oslash$: $\phi(cp_a, D_a^V, \oslash)$
The RBridge path can be assigned to the container cp_a. Then D_a^R becomes $D_a^R = \{q\}$. This is possible if cp_a is not *recursive*.

$$z_{n_1+n_2+n_3+i, n_1+n_2+j} = \begin{cases} \mu(\phi) & \text{if } \ c^1 \neq c^2 \\ \infty & \text{otherwise.} \end{cases} \quad (36)$$

Case 3): The Kit is as $\phi(cp_a, D_a^V, D_a^R)$. Then D_a^R becomes $D_a^R \cup \{q\}$. The matching cost for this case is (now $q \in D_a^R$):

$$z_{n_1+n_2+n_3+i, n_1+n_2+j} = \mu(\phi). \quad (37)$$

The least cost case is then selected for this block.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2015.2413755, IEEE Transactions on Network and Science Management

REFERENCES

[1] D. Belabed, S. Secci, G. Pujolle, and D. Medhi, "Impact of virtual bridging on virtual machine placement in data center networking," in *Proc. of 26th Int. Teletraffic Congress (ITC 2014)*, Sep. 2014.

[2] ——, "Impact of ethernet multipath routing on data center network consolidations," in *Data Center Performance Workshop, Proc. of ICDCS*, 2014.

[3] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, 2008, pp. 63–74.

[4] C. Guo and et al., "Dcell: a scalable and fault-tolerant network structure for data centers," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 75–86.

[5] 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.

[6] J. W. Jiang, T. Lan, S. Ha, M. Chen, and M. Chiang, "Joint vm placement and routing for data center traffic engineering," in *Proc. of IEEE INFOCOM*, 2012.

[7] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Computer Networks*, vol. 57, no. 1, pp. 179–196, 2013.

[8] H. N. Van, F. D. Tran, and J.-M. Menaud, "Performance and power management for cloud infrastructures," in *Proc. of IEEE CLOUD*, 2010.

[9] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, and J. Lawall, "Entropy: a consolidation manager for clusters," in *ACM Int. conference on Virtual execution environments*, 2009.

[10] "Data center architecture overview," in *Cisco Data Center Infrastructure 2.5 Design Guide. OL-11565-01*. Cisco, 2011, pp. 7–16.

[11] R. Perlman, "An algorithm for distributed computation of a spanningtree in an extended lan," in *ACM SIGCOMM Computer Communication Review*, vol. 15, no. 4. ACM, 1985, pp. 44–53.

[12] D. Santos, A. de Sousa, F. Alvelos, M. Dzida, and M. Pióro, "Optimization of link load balancing in multiple spanning tree routing networks," *Telecommunication Systems*, vol. 48, no. 1-2, pp. 109–124, 2011.

[13] "Multi-chassis etherchannel (mec)," Cisco, pp. 7–16, 2011.

[14] "IEEE standard for local and metropolitan area networks - virtual bridged local area networks amendment 10: Provider backbone bridge traffic engineering," Aug 2009.

[15] N. McKeown and et al., "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[16] D. Papadimitriou, E. Dotaro, and M. Vigoureux, "Ethernet layer 2 label switched paths," in *Proc. of Next Generation Internet (NGI), 2005*, 2005.

[17] M. Seaman, "EEE 802.1aq shortest path bridging," *IEEE std*, 2006.

[18] J. Touch and R. Perlman, "Transparent interconnection of lots of links (TRILL): Problem and applicability statement," *RFC 5556*, 2009.

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

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

[21] M. G. Rabbani, R. P. Esteves, M. Podlesny, G. Simon, L. Z. Granville, and R. Boutaba, "On tackling virtual data center embedding problem," in *IFIP/IEEE IM*, 2013.

[22] H. Jin, T. Cheocherngngarn, D. Levy, A. Smith, D. Pan, J. Liu, and N. Pissinou, "Joint host-network optimization for energy-efficient data center networking," *IEEE IPDPS, Boston, MA*, 2013.

[23] O. Biran and et al., "A Stable Network-Aware VM Placement for Cloud Systems," in *Proc. of IEEE/ACM CCGrid 2012*, 2012.

[24] M. Pióro and D. Medhi, *Routing, flow, and capacity design in communication and computer networks*. Elsevier/Morgan Kaufmann, 2004.

[25] M. Balinski, "On finding integer solutions to linear programs," in *Mathematica*, May 1964.

[26] M. Rönnqvist, S. Tragantalerngsak, and J. Holt, "A repeated matching heuristic for the single-source capacitated facility location problem," *European Journal of Operational Research*, vol. 116, pp. 51–68, 1999.

[27] M. Rönnqvist, K. Holmberg, and D. Yuan, "An exact algorithm for the capacitated facility location problems with single sourcing," *European Journal of Operational Research*, vol. 113, pp. 544–559, 1999.

[28] A. Reinert, B. Sansò, and S. Secci, "Design optimization of the petaweb architecture," *Networking, IEEE/ACM Transactions on*, vol. 17, no. 1, pp. 332–345, 2009.

[29] M. Forbes, J. Holt, P. Kilby, and A. Watts, "A matching algorithm with application to bus operations," *Australian Journal of Combinatorics*, vol. 4, pp. 71–85, 1991.

[30] M. Engquist, "A successive shortest path algorithm for the assignment problem," in *INFOR*, 1982, vol. 20, pp. 370–384.

[31] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, pp. 325–340, 1986.

[32] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *ACM SIGCOMM Computer Comm. Review*, vol. 40, no. 1, pp. 92–99, 2010.

[33] T. Benson, A. Akella, and D. Maltz, "Network traffic characteristics of data centers in the wild," in *ACM IMC*, 2010.

[34] A. Greenberg and et al., "Vl2: a scalable and flexible data center network," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 51–62, 2009.

Dallal Belabed is a Ph.D. student at the University Pierre and Marie Curie (UPMC - Paris VI, Sorbonne Universites), France and obtained her M.Sc. from the same university in 2011. In 2011, she worked as a research intern at Alcatel-Lucent France. She also was a visiting researcher at the University of Missouri - Kansas City (USA), in 2012, and at Politecnico di Milano, in 2014. Her interests are about data center network optimization and network function virtualization. http://www-phare.lip6.fr/~belabed.

Stefano Secci is an Associate Professor at the University Pierre and Marie Curie (UPMC - Paris VI, Sorbonne Universites). He received a "Laurea" degree in Telecommunications Engineering from Politecnico di Milano, in 2005, and a dual Ph.D. degree in computer networks from the same school and Telecom ParisTech, in 2009. He also worked as a Research Fellow at NTNU, George Mason University, Ecole Polytechnique de Montreal, and Politecnico di Milano, and as a Network Engineer with Fastweb Italia. His works mostly cover network optimization, protocol design, Internet routing and traffic engineering. His current research interests are about Internet resiliency and Cloud networking. http://lip6.fr/Stefano.Secci.

Guy Pujolle received his the Ph.D. and 'Thèse d'Etat 'degrees in Computer Science from the University of Paris IX and Paris XI, on 1975 and 1978, respectively. He is currently a Professor at the UPMC, and a member of the Institut Universitaire de France. He spent the period 1994-2000 as Professor and Head of the computer science department of Versailles University. He was also Professor and Head of the MASI Laboratory (Pierre et Marie Curie University), 1981-1993, Professor at ENST (Ecole Nationale Superieure des Telecommunications), 1979-1981, and member of the scientific staff of INRIA, 1974-1979.

Deep Medhi is Curators' Professor in the Department of Computer Science and Electrical Engineering at the University of Missouri-Kansas City, USA. He received a B.Sc. (Hons) in Mathematics from Cotton College, Gauhati University, India, his M.Sc. in Mathematics from the University of Delhi, India, and his Ph.D. in Computer Sciences from the University of Wisconsin-Madison, USA. Prior to joining UMKC in 1989, he was a member of the technical staff at AT&T Bell Laboratories. He was an invited Visiting Professor at the Technical University of Denmark, at the University Pierre and Marie Curie, France, and a visiting research fellow at Lund Institute of Technology, Sweden. He was a Fulbright senior specialist during 2003-2008. He served as the chair of the Internet Technical Committee during 2007-2011. He is the editor-in-chief of Springer Journal of Network and Systems Management. His research interests are resilient multi-layer network design, network routing, design and management, next generation networking. He has published extensively, and is co-author of the books, Routing, Flow, and Capacity Design in Communication and Computer Networks (2004) and Network Routing: Algorithms, Protocols, and Architectures (2007), both published by Morgan Kaufmann Publishers (an imprint of Elsevier Science).