Impact of Virtual Bridging on Virtual Machine Placement in Data Center Networking

Dallal Belabed, Stefano Secci, Guy Pujolle, Deep Medhi*
Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris, France. Email: firstname.lastname@upmc.fr
* U. Missouri-Kansas City, 5100 Rockhill Road, Kansas City, MO 64110-2499 USA. Email: dmedhi@umkc.edu

Abstract—The increasing adoption of virtualization techniques has recently favored the emergence of useful switching functions at the hypervisor level, commonly referred to as virtual bridging. In the context of data center network (DCN) consolidations, for VMs colocated in the same virtualization server, virtual bridging becomes very useful to offload inter-VM traffic from access and aggregation switches, at the expense of an additional computing load. DCN consolidations typically chase traffic engineering (TE) and energy efficiency (EE) objectives, and both should be affected by virtual bridging, but it is not intuitive to assert whether virtual bridging acts positively or negatively with respect to TE and EE that should also depend on the DCN topology and forwarding techniques. In this paper, we bring additional understanding about the impact of virtual bridging on DCN consolidations. First, we present a repeated matching heuristic for the generic multi-objective DCN optimization problem, with possible multipath and virtual bridging capabilities, accounting for both TE and EE objectives. Second, we assess the impact of virtual bridging on TE and EE in DCN consolidations. Extensive simulations show us that enabling virtual bridging has a negative impact when EE is the goal and multipath forwarding is adopted, while it leads to important gains, halving the maximum link utilization, when TE is the DCN consolidation goal.

I. INTRODUCTION

The recent achievement of x86 virtualization by advanced software techniques allows attaining virtualization of server and network functions at competitive performance-cost trade-offs with respect to legacy solutions. The increasing adoption of virtualization techniques has recently favored the emergence of useful switching functions at the hypervisor level, commonly referred to as virtual bridging. In the context of data center network (DCN) consolidations, for VMs colocated in the same virtualization server, virtual bridging becomes very useful to offload inter-VM traffic from access and aggregation switches, at the expense of an additional computing load on the physical server. DCN consolidations typically chase traffic engineering (TE) and energy efficiency (EE) objectives, and both should be affected by virtual bridging, but it is not intuitive to assert whether virtual bridging acts positively or negatively with respect to TE and EE that should also depend on the DCN topology and forwarding techniques. In this paper, we bring additional understanding about the impact of virtual bridging on DCN consolidations.

The literature on DCN consolidation problems is extensive. Often referred to as DCN optimization, VM placement or virtual network embedding, the various propositions at the state of the art often have a narrow scope, with a set of constraints so that it is not possible to jointly adopt EE and TE objectives and model advanced multipath forwarding protocols [1]–[3], network link states optimization [4] and edge virtual bridging [5]. In particular, virtual bridging functionalities are rarely modeled, probably because they are marginally understood.

The impact of multipath forwarding protocols on DCN performance strictly depends on the DCN topology. While the most common legacy topology is the 3-tier [6] architecture, it is losing interest because with network virtualization inter-rack traffic in support of consolidation procedures is overcoming the amount of external traffic. Therefore, flats topologies become more interesting. Topologies such as fat-tree [7], DCell [8] and BCube [9] are gaining momentum. The flat nature of these topologies can also give virtual bridging a higher importance in the DCN interconnect. Our previous study investigated the impact of multipath forwarding on different topologies [10]; results showed that multipath forwarding is indeed effective with flattened topology, and that it can be counterproductive when the EE is the primary goal of DCN consolidations (without virtual bridging capabilities).

Furthermore, link states can be monitored when planning Virtual Machines (VMs) migrations. For instance, migrating a VM catalyst of significant traffic at a server (“VM container”) whose access link is close to saturation is not a wise decision. VM containers that are topologically attractive could therefore be favored when deciding where to host and migrate VMs. Commercial DCN consolidation tools (e.g., VMware Capacity Planner [11], IBM CloudBurst [12]), typically are aware of CPU, memory, storage, and energy constraints of VM containers are not, however, aware of link states since the legacy hypothesis is to consider unlimited link capacity. With the emergence of network virtualization, related storage synchronization tools and pervasive virtual bridging, the hypothesis that DCN links have infinite capacity is today becoming inappropriate, especially for DCs facing capital expenditure limitations. Performing VM consolidation that is aware of both container and link states is, however, known to be NP-hard [4]. The complexity does naturally increase when considering multipath and virtual bridging capabilities.

In this context, the recent introduction and large deployment of virtual bridging in most hypervisor solutions (e.g., Xen, KVM, VM Ware NSX), is introducing novel constraints as it becomes interesting to assign to a same container or nearby containers, VMs exchanging large traffic amounts. The impact of virtual bridging on DCN consolidations can be sensible,
depending on topology and forwarding situations as well as on DCN objectives. The contribution of this paper is twofold:

- We describe a virtual machine placement optimization problem in DCN, with multipath forwarding and virtual bridging capabilities, meeting TE and EE objectives, in a novel, compact and versatile formulation. We design a repeated matching heuristic scaling with large DCN sizes. To the best of our knowledge, this is the first comprehensive formulation in this sense, i.e., the first allowing for the consideration of virtual bridging and multipath capabilities and of TE and EE objectives.
- We run our heuristic on realistic DCN consolidation instances, showing that enabling virtual bridging improves TE performance on one hand by roughly two times, and on the other hand it can be counterproductive when EE is the primary goal. With respect to the coexistence of virtual bridging and multipath forwarding, we determine under which circumstances the gain brought by both innovations can be positive.

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

II. BACKGROUND

We briefly discuss state of the art work on Ethernet routing, DCN traffic, and consolidation models.

A. Data Center topologies

The 3-tier architecture [6] is the common legacy of DC topology. It has three layers: access, aggregation and core layers. 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 VLAN and STP switching [13], which, while simple and fast, is known to underutilize the network resources. Even innovations can be positive.

As briefly mentioned in the introduction, alternative topologies have been proposed in recent years. Originally, the authors in [7] proposed a special instance of a Clos topology called "fat-tree" to interconnect commodity Ethernet switches as $k$-ary fat-tree. As depicted in Fig. 1a, 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. 1a shows a fat-tree example for $k = 4$. Another topology is BCube [9], a recursive architecture designed for shipping and container-based, modular data center. As depicted in Fig. 1b, the BCube solution has server devices with multiple ports (typically no more than four). Multiple layers of cheap commodity off-the-shelf mini-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$_{k-1}$s 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 switch, 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 having $n^k n$-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 $j$th server ($i \in [0, n^k−1]$) in the $j^{th}$ BCube$_{k−1}$ ($j \in [0, n − 1]$) to the $j^{th}$ level-$k$ switch. The BCube construction guarantees that switches only connect to servers and never connect directly to other switches, thus multipathing between switches is impossible. It is worth noting that this kind of architecture requires virtual bridging in containers to operate. Fig. 1b shows an example of a BCube$_1$, with $n = 4$.

Similarly to BCube, DCell [8] uses servers equipped with multiple network ports and mini-switches to construct its recursive architecture. In DCell, a server is connected to several other servers and a mini-switch. Generally, a high-level DCell is constructed from low-level DCells. The connection between different DCell networks is typically done by using virtual bridging in containers. 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 mini-switch ($n = 4$ for DCell$_0$ in Fig. 1c). All servers in DCell$_0$ are connected to the mini-switch.

In DCell$_1$, each DCell$_0$ is connected to all the other DCell$_0$s with one link; the Fig. 1c shows a DCell$_1$ example. DCell$_1$ has $n + 1 = 5$ DCell$_0$s. DCell connects the $5$ DCell$_0$s 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 $[0, 4]$, respectively. Then two servers with 2-tuples $[i, j]$ 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 mini-switch, and hence to other nodes within its own DCell$_0$. The other connects to a server in another DCell$_0$. In DCell$_1$, each DCell$_0$, if treated as a virtual node, 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$_0$s, as illustrated in Fig. 1c. A DCell$_k$, is constructed in the same way to the above DCell$_1$ construction. The recursive DCCell construction procedure [8] is more complex than the BCube procedure.
B. Ethernet fabric evolution

In the last decade, several evolutions to the legacy Ethernet switching architecture in terms of TE features have occurred. Under the perspective of incremental upgrade of the Ethernet switching architecture to meet TE requirements, we can consider that the Multiple Spanning Tree Protocol [14] has been the first attempt to actively perform TE in a legacy Ethernet switched network running STP and hence suffering from unused links in normal situations. The multiplexing of multiple clients or VLANs into one among several spanning trees can also be optimized as presented in [14]. Then, other protocols trying to solve bottleneck issues along the spanning tree(s) have been standardized, as notably the Link Aggregation Group or the multi-chassis EtherChannel protocols [15], allowing a switch to use multiple links as a single one with respect to the STP control-plane. Eventually, the real bottleneck in performing TE efficiently in an Ethernet switching context being the spanning tree bridging of Ethernet traffic, the STP control-plane has been removed from more recent carrier Ethernet solutions implementable in DCNs, namely: the Provider Backbone Bridges with Traffic Engineering (PBB-TE) [16], where centralized control servers push MAC tables to backbone switches (in a similar philosophy OpenFlow [3] does too); the L2LSP [17] effort suggesting to use the VLAN fields as MPLS label fields: the already mentioned SPB [2] and TRILL [1] protocols where the control-plane is distributed adapting a layer-3 link state routing protocol (ISIS) to work with the Ethernet data-plane. Nodes in this context are no longer simple bridges since they perform a routing function. Hence, in TRILL, as well as in the following, we refer to them as router-bridges (referred to as RBridges, or RBs).

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, these protocols 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 one of these protocols.

C. DCN traffic models

At present, little is known about DCN traffic characteristics. This is likely because of the high technology heterogeneity and because of non-disclosure confidentiality reasons. There are, however, a few studies worth being mentioned.

Supposing a legacy 3-tier architecture, the authors in [18], [19] collected information from edge, aggregation, and core devices, finding that the traffic originating from a rack show an ON/OFF behavior following heavy-tailed distributions. It is also important to mention that, as presented in [19], for their case, most of server-originated traffic, 80%, stays within the rack, while between 40% and 90% leaves the rack.

In terms of connection volume, the authors in [20] show that, for their case, more than 90% of transfers had a volume from to 100 MB to 1 GB. They also show that 50% of the time, a VM had approximately 10 concurrent flows, and that at least 5% of the time, a machine had more than 80 concurrent flows.

In [21], the authors study the incoming and outgoing traffic rates for seventeen thousand VMs. Their analysis shows that 80% of the VMs had an average rate less than 800 KBytes/min. However, 4% of them had a rate ten times higher. Moreover, the traffic rate’s standard deviation of 82% of the VMs was lower than or equal to twice the mean.

D. Virtual Machine Placement in Data Center Networks

We review thereafter relevant works on the VM placement problem that take network constraints into consideration.

In [21], the authors propose a VM placement solution considering network resource consumption. They assume that a VM container can be modeled as a composition of CPU-memory slots, where each slot can be allocated to any VM. They consider the number of VMs equal to the number of slots; when the number of slots is higher than the number
of VMs, they add dummy VMs (with no traffic), so that the algorithm is not affected. Due to a communication cost between slots, defined as the number of forwarded frames among them, the objective is set as the minimization of the average forwarding latency. They also assume static single-path routing and focus on two traffic models. A dense one where each VM sent traffic to every VMs at an equal and constant rate, and a sparse Infrastructure as a Service (IaaS)-like one with isolated clusters so that only VMs in the same IaaS communicate.

In [22], the authors tackle the problem as a stochastic bin-packing problem, with non-deterministic demands following normal distributions, and propose an ad-hoc heuristic approach to resolve it. In [23], the authors consider network constraints in addition to CPU and memory constraints in the VM placement problem. They defined a network-aware approach to allocate VM placement while satisfying predicted traffic patterns and reducing the worst case cut load ratio in order to support time-varying traffic (they partitioned the set of hosts into non-empty connected subsets, which are bottlenecks for the traffic demand between VMs placed in different sides of the cut).

In [24], the authors revisit the virtual embedding problem by distinguishing between server and bridge nodes with respect to the common formulation. They propose an iterative 3-step heuristic: (i) an arbitrary VM mapping is done; (ii) virtual bridges are mapped to bridges nodes; (iii) virtual links are mapped accordingly. If one of these steps fail, the heuristic comes back to the previous step until a solution is found. The quality of the solution appears very dependent on the first step, the other steps just minimizing the impact of the previous step. Further, there may be scalability issues due to the uncontrollable backtracking. More generally, virtual embedding approaches in the literature often discard specificities of the network control-plane such as the routing protocol and TE capabilities.

In [25], the authors optimize jobs placement, where each job required a number of VMs; the objective function minimizes the network and the node costs. The authors do not handle link capacity constraints, and do not consider multipath forwarding capabilities (instead, multipath routing with one single egress path). In [26], the authors minimize the power energy consumption of activated servers, bridges and links, to maximize the global energy saving. The authors convert the VM placement problems into a routing problem, and so they address the network and server optimization problem as a single one. So, there is no trade-off between the network-side and server-side optimization objective.

Some of these studies ignore link capacity constraints, others exclude dynamic routing as in [21], or just consider the traffic volume to reduce the number of containers as in [22], or just the network resources as in [21] and [23], only [26] consider multipath forwarding capabilities. 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 analyzes virtual bridging impact on VM placement optimization and TE objective considering multipath forwarding.

### III. OPTIMIZATION PROBLEM

We present our optimization problem, describing the constraints in the case where multipath forwarding and virtual bridging are not enabled, then showing how they can be easily extended to enable these features. The notations are provided in Table I. First, we present integrity constraints, then capacity constraints and the objective function, and finally, we position the formulation with respect to the state of the art.

The objective is to balance between maximum link utilization and the number of containers to be activated.

\[
\text{minimize} \quad \alpha U + (1 - \alpha) \sum_{c \in C} b_c \tag{1}
\]

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

\[
\sum_{c \in C} v_{v,c} = 1; \quad \forall v \in V \tag{2}
\]

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

\[
b_c \leq \sum_{v \in V} v_{v,c}; \quad b_c \geq v_{v,c}; \quad \forall c \in C, \forall v \in V \tag{3}
\]

Each container is assigned to one RB:

\[
\sum_{c \in C} v_{c,r} = 1; \quad \forall c \in C \tag{4}
\]

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

\[
\sum_{k} d_{k,r_d}^k = 1 \quad \forall (r_s, r_d) \in R_a \times R_a \tag{5}
\]
A VM is assigned to a container only if there are available residual computing resources:
\[
\sum_{v \in V} d^P_{v,e} e_{v,e} \leq K^P_c \quad \forall c \in C \\
\sum_{v \in V} d^M_{v,e} e_{v,e} \leq K^M_c \quad \forall c \in C
\]  
(6)

Container-RB traffic is less than the access link capacity:
\[
t_{c,r} \leq K_{c,r}; \quad \forall c \in C \quad \forall r \in R
\]  
(7)

Inter-RB traffic is less than the aggregation-core link capacity:
\[
\sum_{r,s} t_{r,s} d^P_{r,s} q^P_{r,s} p^P_{r,s} \leq U_K \quad \forall (r,s) \in T^R
\]  
(8)

Where:
\[
t_{c,r} = \sum_{(v_i,v_j) \in T^V} (t_{v_i,v_j} + t_{v_j,v_i}) e_{v_i,c} e_{v_j,r}; \quad \forall r \in R, \forall c \in C
\]

We have 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 \( \alpha \) factor to assess the trade-off between the two goals and its impact on VM placement and DCN performance.

1) Enabling multipath capabilities: Multipath forwarding between containers and RBs (in the place of LAG, link bonding, or similar approaches) can simply be enabled by declaring \( a_{c,r} \) as a non-negative real variable instead of a binary variable. Hence (4) becomes an integrity constraint on the sum of traffic ratios for each active container to its RBs. Similarly, a multipath between RBs can simply be enabled by declaring \( q^P_{k,d} \) as a non-negative real variable instead of a binary variable. Hence (5) becomes an integrity constraint on the sum of traffic ratios for each pair of RBs to its used paths.

2) Enabling virtual bridging: Enabling virtual bridging means that the container absorbs 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, as a function of the traffic load, is included.

The provided optimization model is an extension of the baseline multi-commodity flow (MCF) problem for network routing with link capacity constraints [27], taking into account: peculiar data center networking constraints due to VM mobility, VM container switching on and off, virtual bridging, and multipath forwarding. In order to control the MCF complexity when handling TE and multipath parameters and variables, we adopted above the link-path formulation [27].

Given the elasticity related to VM migrations and multipathing, requiring double mapping between VMS and VM containers, and between VM containers and usable paths, our optimization problem defined by (2)-(1) even if comprehensive and versatile (considering both unipath and multipath modes, with and without virtual bridging, and VM attachment constraints) is a non-linear problem and cannot be linearized.

IV. HEURISTIC APPROACH

Classically, mapping problems can be revisited as facility location problems, and when capacity constraints need to be verified as a function of the type of mapping, there are similarities with the capacitated facility location problem [28] and, in particular, with the single source facility location problem (SSFLP) [29], [30]. It is easy to derive that our DCN optimization problem can be reduced to the SSFLP and hence is NP-hard. Recently, modeling an optical network dimensioning problem as a facility location problem, the authors in [31] extended a primitive repeated matching heuristic described in [29], [30] to solve the SSFLP and proved it can reach optimality gaps below 5% also for many instances of the problem. A similar approach was later adopted for an optical network dimensioning approach, also providing outstanding performance for very large instances as described in [31].

Motivated by those results, we redesign the repeated matching heuristic to our DCN optimization problem. Nevertheless, the double mapping we have handled in our problem and the multiple capacity constraints to care about (at both link and server sides) make this problem much more difficult to solve, so that the comparison to the optimum is not possible differently than in previous applications [29]–[31].

A. Reformulation of the optimization problem

Recall that DCN communications are between VMs that can be hosted behind the same VM container or behind distant containers interconnected by a DCN path. Certainly, external communications can be modeled introducing fictitious VMs and VM containers acting as egress point, from a functional standpoint. When multipath is enabled, multiple paths can be used, 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.

Let a virtual node be designated by \( v \), \( v \in V \), and 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 that \( D^C \subseteq T^C \). 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. Kit \( \phi \)

A Kit \( \phi \) is composed of a subset of VMs \( D^V \), a VM container pair \( cp \in T^C \) and a subset of RB paths \( D^R \). In a Kit \( \phi \), each VM \( v \in D^V \) is assigned to one of the containers in a pair \( cp (c^i,c^j) \). 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 \( \phi(cp,D^V,D^R) \).
B. Matching Problem

Definition IV.2. Feasible Kit

A Kit $\phi(cp, D^V, D^R)$ is said to be feasible if:

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

Definition IV.3. $L_1$, $L_2$, $L_3$, and $L_4$

$L_1$ is the set of VMs not matched with a container pair. $L_2$ is the set of VM container pairs not matched with an RB path. $L_3$ is the set of RB paths not matched with a container pair. $L_4$ is the set of Kits.

Definition IV.4. Packing $\Pi$

A Packing is a union of Kits in $L_4$. A Packing is said to be feasible if its Kits are feasible and $L_1$ is empty.

B. Matching Problem

Given the DCN optimization problem’s elements using the above described sets, it can be reformulated as a matching problem between them. 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 $s_{i,j} = s_{j,i}$. We introduce the binary variable $z_{i,j}$ that is equal to 1 if $h_i$ is matched with $h_j$ and zero otherwise. The matching problem consists in finding the matching over $A$ that minimizes the total cost of the matched pairs.

$$\min \sum_{i=1}^{q} \sum_{j=1}^{q} s_{i,j} z_{i,j}$$ \hspace{1cm} (9)

s.t. $\sum_{j=1}^{q} z_{i,j} = 1$, $i = 1, \ldots, q$ \hspace{1cm} (10)

$\sum_{i=1}^{q} z_{i,j} = 1$, $j = 1, \ldots, q$ \hspace{1cm} (11)

$z_{i,j} = z_{j,i}$, $i, j = 1, \ldots, q$ \hspace{1cm} (12)

$z_{i,j} \in \{0, 1\}$, $i, j = 1, \ldots, q$ \hspace{1cm} (13)

Selecting the least cost matching vector enables solution improvements via set transformations in 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 infeasible, its cost is set to infinity (a large number in practice). Matching 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, also generating local improvements due to the selection of better VM containers or RB routes; note that for these block local exchange linear optimization 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 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 link capacity constraints as described above does not need to be enforced during the repeated matching iterations, but 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’s fixed energy consumptions is one of the goals of the DC provider. The overall Kit cost is not meant to represent a direct monetary cost, but it is such that the
repeated matching promotes less expensive and more efficient Kits. Therefore, to align with the objective function (1), and remembering that the cost of a Packing corresponds to the cost of its Kits, we set the cost of a Kit \( \phi(c_P, D^V, D^R) \) as:

\[
\mu(\phi) = (1 - \alpha)\mu^E(\phi) + \alpha \mu^T(\phi)
\]

(14)

Where \( \alpha \) is the trade-off scaling factor between the EE and the TE components, that are, respectively:

\[
\mu^E(\phi) = \sum_{\bar{e} \in c_P} \left( \frac{K_{\bar{e}}^P}{\sum_{v \in D^V} d_{v}^{\bar{e}}} + \frac{K_{\bar{e}}^M}{\sum_{v \in D^V} d_{v}^{\bar{e}}} + \Gamma T_v \right)
\]

(15)

\[
\mu^T(\phi) = \max_{(n_i, n_j) \in \Pi} U_{n_i, n_j}(\Pi)
\]

(16)

Where \( T_v \) represents the global traffic \( v \) sends and receives, i.e., \( T_v = \sum_{e' \in V} t_{v,v'} \); \( \Gamma \) is the additional power and memory, to take into account the impact of traffic to VM container’s CPU and memory consumption when virtual bridging is enabled (zero otherwise). Note that the computing capacity constraints are indirectly enforced within the \( L_A L_A \) matching cost computation. \( U_{n_i, n_j}(\Pi) \) is the link utilization of each link used by the current Packing \( \Pi \) solution, so that the maximum link utilization experienced by the Kit’s RB paths can be minimized. In our heuristic, in order to linearly compute the RB paths’ link utilization, the aggregation and core links of RB paths are considered as congestion free, while access container-RB links are considered as prone to congestion. Then generally adheres to the reality of most DCNs today as access links are typically 1 Gbps Ethernet links while aggregation/core links reach the 10 Gbps and 40 Gbps rates. This is a realistic approximation a acceptable in a heuristic approach, especially because it allows a significant decrease in the heuristic’s 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 lower the time complexity. We have implemented the algorithm in [32], based on the method of Engquist [33]. Its starting point is the solution vector of the matching problem without the symmetry constraint (12) obtained with the algorithm described in [34] (which was chosen for its speed performance; its output is a symmetric solution matching vector).

Designing the matching costs in an efficient and rational way, the Packing cost across iterations should be decreasing, monotonically starting by the moment when \( L_1 \) gets empty; moreover, Step 2 should be reached and the heuristic converges, and \( L_1 \) at the last step should be empty.

V. SIMULATION RESULTS

We implemented our heuristic using Matlab, and CPLEX for the matching cost computation of some blocks. The adopted VM containers correspond to an Intel Xeon 5100 server with 2 cores of 2.33GHz and 20GB RAM, able to host 16 VMs. We study the virtual bridging impact on virtual machine placement under two scenarios: when TE is the primary goal and when EE is the primary goal. We also analyze what happens when multipath forwarding is enabled at the bridge level.

We executed our heuristic with the following DCN topologies: 3-tier, fat-tree, BCube and DCell. We note that BCube and DCell work properly only by employing virtual bridging at the server level; we allowed for a small modification of the topology to allow a reference comparison between them and the other topologies for the cases without virtual bridging, calling these variations BCube* and DCell*, respectively. We allow 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.

In the simulations, all DCNs are loaded at 85% in terms of computing and network capacity. Note that with all topologies, we allowed for a certain level of overbooking in the resource allocation for the sake of algorithm fluidity especially at starting and intermediate iterations. The capacity of the access link was set to 1 Gbps. As not all VMs communicate to 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 [21], with clusters of up to 30 VMs communicating with each other and not communicating with other IaaS’s VMs. Within each IaaS, the traffic matrix was built accordingly to the traffic distribution of [20]. We ran 30 different instances with different traffic matrices for each case, and the results reported in the following have a confidence interval of 95%. Our heuristic was fast (reached convergence roughly within a dozen of minutes per execution) and successfully reached a steady state (i.e., three iterations with to the same solution, characterized by a feasible Packing as previously described).

A. Virtual bridging impact under EE-oriented consolidations

Fig. 3 illustrates the results in terms of enabled VM containers for different topologies when EE is the 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. Observing the results we can assess that:

- the impact of virtual bridging in DCN consolidations when the EE is the goal leads to negligible differences in EE performance;
- with multipath forwarding, the use of virtual bridging was counterproductive;
- the DCell topology shows 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, did show the overall worst EE performance for single-path forwarding and better EE performance for multipath forwarding, with negligible difference from each other.

All in all, the main outcome of this analysis is 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.

B. Virtual bridging impact under TE-oriented consolidations

As already mentioned, EE goals can be considered the opposite of TE goals. Chasing EE tends to minimize the number of enabled VM containers, yet no care is given to network link utilization. We rerun the experimentations setting the traffic engineering goal as the DCN consolidation objective, i.e., $\alpha = 1$, considering singlepath and multipath forwarding, for the different topologies. Results are reported in Fig. 4. Observing the results we can assess that:

- virtual bridging always leads to sensible TE performance gains;
- with singlepath forwarding, the DCell 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 indirectly minimizing the number of links used to interconnect servers.
- with multipath forwarding, the BCube gets the largest TE gain, with maximum link utilization being halved from about 80% to 40%.
- BCube and DCell do have similar TE performances, with BCube probably because the gain in path diversity brought by virtual bridging is higher with BCube (which keeps a core layer unlike the DCell);
- the TE gain with respect to hierarchical topologies (Fat-tree, 3-tier) is always positive and slightly higher with enabled multipath forwarding.

These TE performance results are not intuitive and relevant. It is definitely interesting to adopt virtual bridging when the primary goal of DCN consolidations 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.

VI. CONCLUSIONS

Data Center Networking is a challenging field of applications of old and new technologies and concepts. In this paper, we investigated how traffic engineering and EE goals in virtual machine consolidations can coexist with the emergence of virtual bridging, i.e., the capability to switch Ethernet traffic at the hypervisor level in virtualization servers. We also studied such an impact when multipath forwarding is enabled.

We provide a versatile formulation of the virtual machine placement problem supporting virtual bridging capabilities and multipath forwarding, and describe a repeated matching heuristic for its resolution on dense and large topologies.

Moreover, through extensive simulation of realistic instances with legacy and novel flat DC topologies, we discovered that when EE is the primary goal of DCN optimization, virtual bridging can be counterproductive and should not be enabled. When TE is the primary goal instead, the TE performance gain can be very important and 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. The gain with respect to more hierarchical topologies (Fat-tree, 3-tier) is also important.
ACKNOWLEDGMENT

This work was partially supported by the Systematic FUI 15 project RAVIR (http://www.ravir.io) and National Science Foundation grant CNS-0916505.

REFERENCES