Beyond the Stars: Revisiting Virtual Cluster Embeddings

Matthias Rost
TU Berlin, Germany
mrost@inet.tu-berlin.de

Carlo Fuerst
TU Berlin, Germany
carlo@inet.tu-berlin.de

Stefan Schmid
TU Berlin, Germany & T-Labs, Germany
stefan@net.t-labs.tu-berlin.de

ABSTRACT

It is well-known that cloud application performance can critically depend on the network. Over the last years, several systems have been developed which provide the application with the illusion of a *virtual cluster*: a star-shaped virtual network topology connecting virtual machines to a logical switch with absolute bandwidth guarantees.

In this paper, we debunk some of the myths around the virtual cluster embedding problem. First, we show that the virtual cluster embedding problem is not NP-hard, and present the fast and optimal embedding algorithm VC-ACE for arbitrary datacenter topologies. Second, we argue that resources may be wasted by enforcing star-topology embeddings, and alternatively promote a hose embedding approach. We discuss the computational complexity of hose embeddings and derive the HVC-ACE algorithm. Using simulations we substantiate the benefits of hose embeddings in terms of acceptance ratio and the resource footprint.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.3 [**Network Operations**]: Network Management; G.1.6 [**Optimization**]: Linear Programming

Keywords

Network Virtualization, Datacenter, Resource Allocation, Virtual Cluster, Hose Model

1. INTRODUCTION

Cloud applications such as MapReduce and scale-out databases generate large amounts of network traffic, and a considerable fraction of their runtime is due to network activity. For example, traces of MapReduce jobs from a Facebook cluster reveal that network transfers on average account for 33% of the execution time. [5]

The application performance hence critically depends on the underlying network, and without strict bandwidth reservations, the execution can become unpredictable and subject to a high variance. [13] Indeed, it has recently been shown that the bandwidth available to an application can differ by a factor of five or more [17], even within the same day.

Consequently, several systems have been proposed over the last years which allow the application to specify network requirements and construct an inter-connecting virtual network with bandwidth guarantees. [3, 16]

A popular virtual network abstraction introduced by Ballani et al. [3] is the *virtual cluster*: a Virtual Cluster VC($\mathcal{N}, \mathcal{B}, \mathcal{C}$) *guarantees* a specified *minimal* bandwidth $\mathcal{B}$ between the $\mathcal{N}$ many virtual machines (VMs) of *size* $\mathcal{C}$ and a non-oversubscribed *logical switch*, independently of the VM locations in the datacenter topology. A virtual cluster is attractive for its simplicity – it describes a simple star topology – and its flexibility – it supports all communication patterns in which the aggregate ingress and aggregate egress bandwidth at each VM is at most $\mathcal{B}$.

The question of how to embed a virtual cluster in a given datacenter such that it consumes a minimal amount of resources while providing its performance guarantees, is an interesting algorithmic problem. However, so far, only heuristic solutions are known for the virtual cluster embedding problem. Moreover, existing algorithms are usually tailored towards *aggregated* fat tree datacenter topologies: while given today's ECMP control planes, this may make sense, such algorithms cannot exploit the full path diversity present in the network, and also do not apply to modern hypercubic topologies such as BCube [9], Jellyfish [14], MDCube [15].

In fact, sometimes it is even claimed that the problem of "*allocating virtual cluster requests on graphs with bandwidth-constrained edges is NP-hard*" [3], and accordingly, researchers have resorted to weaker quality measures, such as *spatial locality* [16].

Contributions. We revisit the virtual cluster embedding problem and offer two main contributions. First, we present a polynomial-time algorithm VC-ACE[1] to compute resource-minimal virtual cluster embeddings on general (and both uncapacitated and capacitated) substrate topologies, including the frequently studied aggregated fat trees [3, 16], but also non-aggregated fat trees [1] as well as modern hypercubic topologies such as the BCube [9]. This result also implies that the virtual cluster embedding problem as studied in [3, 16] is *not NP-hard*. Indeed, existing literature on the virtual cluster embedding problem often refers to more general embedding problems (e.g., virtual network embedding). Second, we argue that virtual clusters should be understood as *abstractions*, and their actual embeddings do not necessarily have to be star-like: by using a *hose mapping*—i.e., by replacing the logical switch of the virtual cluster by a set of direct interconnections supporting all hose traffic matrices [10]— both the embedding cost (resource footprint) and the acceptance ratio can be significantly improved. We will refer to this embedding variant as the hose virtual cluster, short HVC. We study the computational complexity of HVC embeddings, showing the *NP*-hardness and the impos-

[1] *ACE* stands for "All Centers Embedding", as it iterates through all center possibilities (see Section 3).

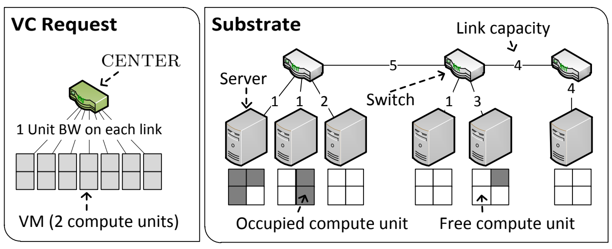

Figure 1: The VC with $\mathcal{N} = 7$, $\mathcal{C} = 2$, $\mathcal{B} = 1$ on the *left* shall be embedded on the substrate on the *right*, such that node and link capacities are respected.

siblity to approximate this type of embedding in capacitated networks. In order to reap the benefits of HVC without sacrificing computational tractability, we propose the efficient *splittable* algorithm HVC-ACE and report on our simulations showing the benefits of the hose model.

2. MODEL

A Virtual Cluster VC($\mathcal{N}$,$\mathcal{B}$,$\mathcal{C}$) provides the abstraction of a virtual network connecting each of the $\mathcal{N} \in \mathbb{N}$ virtual machines (VMs) to a *logical switch* at a bandwidth of at least $\mathcal{B} \in \mathbb{N}$. Each VM needs $\mathcal{C} \in \mathbb{N}$ compute units (e.g., $\mathcal{C}$ CPU cores including memory). Formally, VC($\mathcal{N}$,$\mathcal{B}$,$\mathcal{C}$) is modelled as an undirected graph $\text{VC} = (V_{\text{VC}}, E_{\text{VC}})$ with $V_{\text{VC}} = \{1, 2, \ldots, \mathcal{N}, \textsc{center}\}$ and $E_{\text{VC}} = \{\{i, \textsc{center}\} | 1 \leq i \leq \mathcal{N}\}$, where CENTER denotes the logical switch to which the VMs are connected.

While the embedding of general virtual networks has been studied intensively in the literature (e.g., [4]), the virtual cluster embedding problem has only been considered for fat trees, today's predominant datacenter topology. [1]

This paper considers more general substrate topologies. Formally, we model the substrate S as an undirected graph $\text{S} = (V_{\text{S}}, E_{\text{S}}, \textsc{cap}, \textsc{cost})$ with (currently *available*) node and link capacities $\textsc{cap} : V_{\text{S}} \cup E_{\text{S}} \to \mathbb{N}$. Moreover, we consider a general model where nodes and links may even have different costs $\textsc{cost} : V_{\text{S}} \cup E_{\text{S}} \to \mathbb{R}_{\geq 0}$.

A virtual cluster embedding is a mapping of the virtual cluster's VMs and the logical switch to a physical server in the substrate network together with the allocation of bandwidth $\mathcal{B}$ from each VM to the logical switch. A *valid* embedding does not oversubscribe server and network resources; an *optimal* embedding minimizes the overall cost of the allocated node and link resources (the "resource footprint"). Formally, a feasible embedding of a VC($\mathcal{N}$,$\mathcal{B}$,$\mathcal{C}$) on the substrate network S is a mapping $\textsc{map}_V : V_{\text{VC}} \to V_{\text{S}}$ of each virtual cluster node onto a substrate node together with a mapping of virtual cluster edges onto *paths* in the substrate network $\textsc{map}_E : E_{\text{VC}} \to \mathcal{P}(E_{\text{S}})$, such that the path $\textsc{map}_E(\{u, v\})$ connects $\textsc{map}_V(u)$ and $\textsc{map}_V(v)$ in S for $\{u, v\} \in E_{\text{VC}}$ and capacities are not violated, formally:

$$\sum_{\substack{v' \in V_{\text{VC}} \setminus \{\textsc{center}\} \\ v = \textsc{map}_V(v')}} \mathcal{C} \leq \textsc{cap}(v) \quad \text{and} \quad \sum_{\substack{e' \in E_{\text{VC}} \\ e \in \textsc{map}_E(e')}} \mathcal{B} \leq \textsc{cap}(e) \qquad (1)$$

hold for $v \in V_{\text{S}}, e \in E_{\text{S}}$. The *Virtual Cluster Embedding Problem* [3, 16] asks for an valid embedding of *minimal cost*:

$$\mathcal{C} \cdot \sum_{v \in V_{\text{VC}} \setminus \{\textsc{center}\}} \textsc{cost}(\textsc{map}_V(v)) + \mathcal{B} \cdot \sum_{\substack{e' \in E_{\text{VC}} \\ e \in \textsc{map}_E(e')}} \textsc{cost}(e) \ . \qquad (2)$$

Algorithm 1: The VC-ACE Algorithm

```
  Input:  Substrate $S = (V_S, E_S)$, request VC($\mathcal{N}$,$\mathcal{B}$,$\mathcal{C}$)
  Output: Optimal VC mapping $\textsc{map}_V, \textsc{map}_E$ if feasible
1  $(\hat{f}, \hat{v}) \leftarrow$ (null, null)
2  $V_{S'} = V_S \cup \{s^+\}$ and $E_{S'} = E_S \cup \{(s^+, u) | u \in V_S\}$
3  $\textsc{cap}_{S'}(e) = \begin{cases} \lfloor \textsc{cap}(e)/\mathcal{B} \rfloor & , \text{if } e \in E_S \\ \lfloor \textsc{cap}(u)/\mathcal{C} \rfloor & , \text{if } e = (s^+, u) \in E_{S'} \end{cases}$
4  $\textsc{cost}_{S'}(e) = \begin{cases} \textsc{cost}(e) \cdot \mathcal{B} & , \text{if } e \in E_S \\ \textsc{cost}(u) \cdot \mathcal{C} & , \text{if } e = (s^+, u) \in E_{S'} \end{cases}$
5  for $v \in V_S$ do
6      $f \leftarrow$ MinCostFlow($s^+, v, \mathcal{N}, V_{S'}, E_{S'}, \textsc{cap}_{S'}, \textsc{cost}_{S'}$)
7      if $f$ is feasible and cost($f$) < cost($\hat{f}$) then
8          $(\hat{f}, \hat{v}) \leftarrow (f, v)$
9  if $\hat{f}$ = null then
10     return null
11 return DecomposeFlowIntoMapping($\hat{f}, \hat{v}$)
```

Figure 1 shows an example: the virtual cluster VC on the *left* should be embedded on the physical substrate on the *right*. In this example, each server has a capacity of 4 units (e.g., it has four cores) while VMs require 2 units each ($\mathcal{C} = 2$); some resources are already in use (e.g., by other virtual clusters).

3. OPTIMAL VC EMBEDDINGS

This section presents the fast and optimal algorithm VC-ACE for embedding virtual clusters on general topologies. At the heart of VC-ACE (see Algorithm 1) lies the observation that the virtual embedding problem can be reduced to a series of flow problems on an extended substrate graph (see Figure 2). We exploit the following facts:

1) The required bandwidth $\mathcal{B}$ and the respective compute resources $\mathcal{C}$ of each VM in a virtual cluster is the same. As connections between the VMs and CENTER are embedded as *unsplittable* paths, the substrate's edge capacities (and costs) can be normalized for the case $\mathcal{B} = 1$.

2) Assuming that the VM mappings $\textsc{map}_V$ as well as the location of the center are fixed, the cost-optimal link mapping can be computed in polynomial-time. Concretely, the minimum-cost unsplittable multi-commodity flow problem with commodities $\{(\textsc{map}_V(i), \textsc{map}_V(\textsc{center}), 1)\}$ can be transformed into an integral minimum-cost single-commodity flow problem by introducing a super source s^+ together along edges $e_i^+ = (s^+, \textsc{map}_V(i))$ with $\textsc{cap}(e_i^+) = 1$ for $1 \leq i \leq \mathcal{N}$. It suffices then to ask for an integral minimum-cost flow of value $\mathcal{N}$ from s^+ to $\textsc{map}_V(\textsc{center})$. The equivalence of these problems follows from construction, since $\mathcal{B} = 1$ holds and edge capacities are integral (cf. [12]).

3) Assume that the mapping of CENTER is fixed. The *mapping decision* for the VMs $\{1, \ldots, \mathcal{N}\}$ can be incorporated into the integral minimum-cost flow problem in the following way. The super source s^+ is connected to *all* substrate nodes $u \in V_{\text{S}}$ via $e_u^+ = (s^+, u)$ with $\textsc{cap}(e_u^+) = \lfloor \textsc{cap}(u)/\mathcal{C} \rfloor$ and $\textsc{cost}(e_u^+) = \textsc{cost}(u) \cdot \mathcal{C}$. We

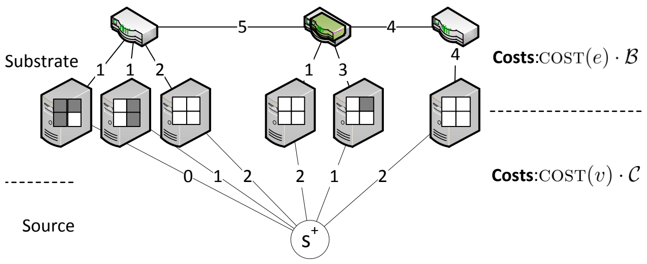
Substrate
Costs:$\text{COST}(e)\cdot\mathcal{B}$
Costs:$\text{COST}(v)\cdot\mathcal{C}$
Source
S^+

Figure 2: The flow problem for the situation from Figure 1, if center is mapped to the central switch.

now consider an integral minimum-cost flow from S^+ to the fixed location of the CENTER, i.e. $\text{MAP}_V(\text{CENTER})$. If such a flow $f : E_S \to \mathbb{N}$ of value $\mathcal{N}$ exists, then $\sum_{u\in V_S} f(e_u^+) = \mathcal{N}$ holds and $f(e_u^+)$ can be identified with the number of VMs that are placed on u. By construction, placing $f(e_u^+) \in \mathbb{N}$ many VMs onto u cannot violate node capacities and costs are correctly accounted for. Lastly, flows may only terminate at $\text{MAP}_V(\text{CENTER})$ and hence each node $u \in V_S$ establishes exactly $f(e_u^+)$ many unsplittable paths to $\text{MAP}_V(\text{CENTER})$.

The above insights are instrumental for designing VC-ACE (see Algorithm 1) and for understanding its correctness. Based on 3), if the virtual switch's mapping is fixed, then the optimal embedding can be computed by solving a single integral minimum-cost flow problem (see Line 7) on a specifically constructed graph (see Lines 2-6). For each possible location of CENTER the optimal flow together with the mapping of CENTER is stored (see Lines 8,9). Lastly, if a feasible flow existed, the cost optimal flow $\hat{f}$ is decomposed into paths $P = \langle S^+, u_1, \ldots, u_n, \text{MAP}_V(\text{CENTER})\rangle$, yielding a VM placement on node u_1 together with the substrate path $\langle u_1, \ldots, u_n, \text{MAP}_V(\text{CENTER})\rangle$ towards $\text{MAP}_V(\text{CENTER})$.

VC-ACE has a polynomial runtime, which is dominated by solving exactly $|V_S|$ many minimum-cost flow problems. By employing the *Successive Shortest Paths Algorithm*, we obtain a runtime of $\mathcal{O}\left(\mathcal{N}(n^2\log n + n\cdot m)\right)$, where $n = |V_S|$ and $m = |E_S|$. On tree topologies like the fat tree, the runtime of VC-ACE can be reduced to $\mathcal{O}(n\cdot\mathcal{N})$, which is on par with the best known heuristic approaches [3, 16].

4. HOSE-BASED VC EMBEDDINGS

A virtual cluster essentially supports any *communication pattern* between the VMs for which the aggregate ingress and aggregate egress bandwidth at each VM is at most $\mathcal{B}$. If the only role of the logical switch in the VC abstraction is to facilitate these communication patterns, it is wasteful to enforce the explicit star embedding and the redirection via the unique center. Thus, one may consider to remove the switch and rather support the communication using *direct interconnections* between VM pairs in a hose fashion [6].

In the following, we introduce the respective *hose embedding* problem and study how to solve it. We define the hose virtual cluster embedding problem and highlight its potential benefits in Section 4.1. We however also show that the unsplittable hose embedding is computationally hard (in Section 4.2), and present an optimal Mixed-Integer Programming (MIP) formulation in Section 4.3, which also forms the basis of our proposed splittable-hose cluster embedding algorithm HVC-ACE presented in Section 4.4.

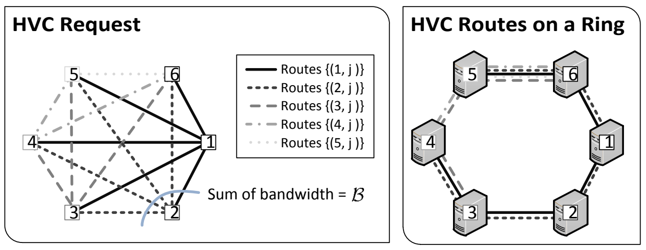

Figure 3: A hose-based virtual cluster HVC=(6,1,1) *(left)* and a feasible embedding of the HVC on a 6-node substrate ring *(right)* with 2 units of bandwidth on each link and one compute unit on each server.

4.1 Problem Definition and Motivation

Henceforth, we denote by $V_C = \{1, \ldots, \mathcal{N}\}$ the set of VMs and by $E_C = \{(i,j) | i,j \in V_C, i < j\}$ the set of interconnections. A feasible solution to the hose-based virtual cluster embedding problem is characterized as follows:

1. The mapping of VMs must not violate node capacities (cf. Equation 1).
2. *Routes* $(i,j) \in E_C$ are realized as *simple* paths $\text{MAP}_E(\{i,j\}) \subseteq E_S$, connecting $\text{MAP}_V(i)$ and $\text{MAP}_V(j)$.
3. The (oblivious) routing according to MAP_E does not violate the substrate's edge capacities *under any communication pattern.* Concretely, there exists an integral bandwidth reservation $l_{u,v} \leq \text{CAP}(u,v)$ for $\{u,v\} \in E_S$, such that for all feasible traffic matrices M_{ij}, i.e., for $i \in V_C$ it holds $\sum_{(j,i)\in E_C} M_{ji} + M_{ij} \leq \mathcal{B}$, and the bandwidth reservation is not exceeded:

$$\sum_{\{i,j\}\in E_C : \{u,v\}\in \text{MAP}_E(\{i,j\})} M_{ij} \leq l_{u,v}\,.$$

The cost of a *hose-based* virtual cluster embedding, given bandwidth reservations $l_{u,v}$ is defined analogously to Equation 2:

$$\mathcal{C}\cdot\sum_{i\in V_C}\text{COST}(\text{MAP}_V(i)) + \mathcal{B}\cdot\sum_{e\in E_S} l_{u,v}\cdot\text{COST}(e).$$

In the following, we will refer to this virtual cluster interpretation omitting the logical switch as the *hose-based virtual cluster*, short HVC.

In order to clarify and highlight the difference between the two virtual cluster interpretations, we consider the example in Figure 3: a ring substrate with 6 nodes, where nodes have a capacity of one unit and links have a capacity of two units, and assume the virtual cluster with $\mathcal{N} = 6, \mathcal{B} = 1, \mathcal{C} = 1$.

First we observe that it is impossible to map a logical switch CENTER in this scenario: each ring node must host one VM and any placement of the CENTER therefore requires the establishment of $\mathcal{N} - 1 = 5$ many independent paths to the respective substrate node onto which CENTER is mapped. This is impossible, since each node's accumulated bandwidth is 4; no "classic" (star) VC embedding exists.

In the hose-based virtual cluster HVC however, a feasible embedding (depicted in Figure 3) can be computed. To see that this is indeed a feasible solution, consider e.g., the substrate edge e connecting $\text{MAP}_V(5)$ and $\text{MAP}_V(6)$. The edge e lies on the routing paths of the VM pairs $R(e) =$

$\{(1,5),(2,5),(3,6),(4,6),(5,6)\}$. Despite e's capacity being 2, this still is a feasible solution. The load on e amounts to $\sum_{(i,j)\in R(e)} M_{ij}$ for a traffic matrix M_{ij}. As M_{ij} is required to respect the *cumulative* bandwidth $\mathcal{B}$, this load is bounded by $M_{1,5} + M_{2,5} \leq 1$ and $M_{3,6} + M_{4,6} + M_{5,6} \leq 1$ and the maximally achievable load l_e is indeed $2 \leq \textsc{cap}(e)$.

This example highlights a *qualitative* disparity, in the sense that only HVCs can be embedded. Analogously, it is easy to construct examples where both VC and HVC can be embedded, but the corresponding optimal embeddings differ significantly in their costs. For instance, consider the above ring network with an additional node u whose compute capacity is 0, and which connects to all ring nodes with edges of bandwidth 1 at some cost $c > 0$. While the HVC solution still has the same cost, the cost of the unique VC solution is $\mathcal{N} \cdot c$ which can be arbitrarily high. Our computational evaluation (see Section 5) shows that this qualitative disparity also arises in other topologies, e.g. in fat trees.

4.2 Computational Complexity

The above example has shown the potential benefit of using the hose model. But what is the computational complexity of hose embeddings? In the following, we show that an optimal hose embedding is actually a star embedding, if edge capacities can be neglected. If edge capacities cannot be neglected, the respective embedding problem is hard and even inapproximable, unless $P = NP$ holds.

Case I: Bandwidth Requirements Are Negligible.

We first present the perhaps surprising result, that VC and HVC embeddings are the same, if enough bandwidth is available, concretely if $\textsc{cap}(e) \geq \mathcal{N} \cdot \mathcal{B}$ holds for all edges $e \in E_{\text{S}}$. This is a non-trivial result which follows from the famous VPN conjecture (proven by Goyal et al. [7]). In a nutshell, the VPN conjecture states that in *uncapacitated networks*, hose embedding problems with symmetric bandwidth bounds and no restrictions on routing (known as the `SymG` model), can be reduced to hose problem instances in which routing paths *must form a tree* (known as the `SymT` model). Based on this result, any optimal HVC embedding will have a central 'hub' (corresponding to the VC's logical switch) in the network, such that all traffic passes this node (cf. [7]).

Thus, when not considering link capacities any optimal HVC embedding contains a central 'hub'. Recall algorithm VC-ACE presented in Section 3 explicitly computes the *minimum cost* embedding towards such a node, called $\textsc{center}$. By construction of the $\mathcal{N}$ flows towards the center carrying exactly $\mathcal{B}$ much bandwidth, no VC solution will use more than $\mathcal{N} \cdot \mathcal{B}$ bandwidth and thus by the VPN conjecture, the equality of VC and HVC solutions follows.

Case II: Bandwidth Requirements Are Not Negligible.

We will now show that the HVC embedding problem is generally – on non-tree topologies – inapproximable unless $P = NP$ holds. This result again follows from the literature on hose embedding problems, where terminals are fixed. In particular, it was proven in [10] that the capacitated hose model is *NP*-hard. Even deciding whether a solution exists is *NP*-complete, implying that – unless $P = NP$ holds – no polynomial-time approximation algorithm exists.

A simple reduction shows that this result translates to the HVC embedding problem: Given is a symmetric hose problem on $(V_{\text{S}}, E_{\text{S}})$ with the set of terminals $T \subseteq V_{\text{S}}$ and integral bandwidth bounds $b_t \in \mathbb{N}$ for $t \in T$. By defining $\textsc{cap}(t) = b_t$ and $\textsc{cost}(t) = 0$ for all $t \in T$, the original hose problem is equivalent to the respective HVC problem of embedding the virtual cluster $\text{VC}(\sum_{t\in T} b_t, 1, 1)$ onto the substrate network with edge costs and capacities remaining the same.

4.3 Exact Unsplittable Hose Algorithm

The above inapproximability result rules out any type of efficient approximation algorithms. Hence we give the exact Mixed-Integer Programming formulation HVC-OSPE for obtaining *Optimal* Single-Path Embeddings.

Our formulation (see MIP 1) builds upon the compact hose formulation by Altin et al. [2]. We employ the following additional notation. We use $E_{\overleftrightarrow{S}} = \{(u,v),(v,u) | \{u,v\} \in E_{\text{S}}\}$ to denote the set of bi-directed substrate edges, and we denote by $\delta^-_u = \{(u,v) \in E_{\overleftrightarrow{S}}\}$ and $\delta^+_u = \{(v,u) \in E_{\overleftrightarrow{S}}\}$ the incoming and outgoing (directed) edges of $u \in V_{\text{S}}$ respectively. We generalize the δ^+ notation also for sets and use $\delta^+(W) = \{(u,v) | u \in W, v \notin W\}$. For breaking symmetries, we assume some arbitrary numbering of the VMs, given by the bijection $\sigma : V_{\text{S}} \to \{1, \ldots, |V_{\text{S}}|\}$.

We introduce node mapping variables $x^i_u \in \{0,1\}$, with $x^i_u = 1$ iff. $i \in V_C$ is mapped onto the substrate node $u \in V_{\text{S}}$. Given these mapping variables, the symmetric hose formulation of [2] can be adapted: The routing variables $y^{ij}_{uv} \in \{0,1\}$ for $(i,j) \in E_C$ and $(u,v) \in E_{\overleftrightarrow{S}}$ determine the simple path between *the substrate nodes* onto which i and j have been mapped (see Constraint 8). The cumulative load variable of a substrate edge $\{u,v\}$ is introduced as $l_{uv} \in \mathbb{N}$. The load variables are lower bounded according to the 'dual' variables $\omega^s_{uv} \geq 0$ for all $s \in V_C, \{u,v\} \in E_{\text{S}}$ (see Constraints 9 and 10, and [2] for an in-depth explanation).

With respect to the node mapping, Constraint 4 enforces that each virtual cluster node is mapped onto exactly one substrate node. Constraints 6 and 7 bound the resource allocations in the substrate by the respective node and edge capacities. Lastly, we introduce Constraint 5 to *break symmetries*: as all nodes in V_C have identical resource requirements, a feasible node mapping induces up to $\mathcal{N}!$ equivalent ones. Constraint 5 only allows for one of these permutations.

Mixed-Integer Program 1: HVC-OSPE

$$\min \sum_{i\in V_C, u\in V_{\text{S}}} \textsc{cost}_u \cdot x^i_u + \sum_{\{u,v\}\in E_{\text{S}}} \textsc{cost}_{u,v} \cdot l_{uv} \tag{3}$$

$$\sum_{u\in V_{\text{S}}} x^i_u = 1 \qquad \forall i \in V_C. \tag{4}$$

$$\sum_{u\in V_{\text{S}}} \sigma_u \cdot (x^i_u - x^{i+1}_u) \leq 0 \qquad \forall i \in V_C \setminus \{\mathcal{N}\}. \tag{5}$$

$$\sum_{i\in V_C} \mathcal{C} \cdot x^i_u \leq \textsc{cap}_u \qquad \forall u \in V_{\text{S}}. \tag{6}$$

$$l_{uv} \leq \textsc{cap}_{uv} \qquad \forall \{u,v\} \in E_{\text{S}}. \tag{7}$$

$$\sum_{(u,v)\in\delta^+_u} y^{ij}_{uv} - \sum_{(v,u)\in\delta^-_u} y^{ij}_{vu} = x^i_u - x^j_u \qquad \forall (i,j) \in E_C,\ \forall u \in V_{\text{S}}. \tag{8}$$

$$\sum_{i\in V_C} \mathcal{B} \cdot \omega^i_{uv} \leq l_{uv} \qquad \forall \{u,v\} \in E_{\text{S}}. \tag{9}$$

$$y^{ij}_{uv} + y^{ij}_{vu} \leq \omega^i_{uv} + \omega^j_{uv} \qquad \forall (i,j) \in E_C,\ \forall \{u,v\} \in E_{\text{S}}. \tag{10}$$

Linear Program 2: HMPR

$$\min \sum_{\{u,v\}\in E_S} \text{COST}_{u,v} \cdot l_{uv} \quad (11)$$

$$l_{uv} \le \text{CAP}_{uv} \quad \forall \{u,v\} \in E_S. \quad (12)$$

$$\sum_{i\in V_C} \mathcal{B} \cdot \omega^i_{uv} \le l_{uv} \quad \forall \{u,v\} \in E_S. \quad (13)$$

$$\sum_{(u,v)\in\delta^+(W)} (\omega^i_{uv} + \omega^j_{uv}) \ge 1 \quad \forall (i,j) \in E_{VC}. \forall W \subset V_S : \text{MAP}_V(i) \in W, \text{MAP}_V(j) \notin W \quad (14)$$

Despite breaking symmetries the formulation HVC-OSPE remains hardly solvable even for small networks. Our initial computational experiments have shown that the mean runtime of HVC-OSPE for optimally embedding a 10-node VC onto a 20-node substrate already exceeds 10 minutes. One reason for this runtime can be found in the number of integral flow variables which amounts to $\Theta(|E_S| \cdot \mathcal{N}^2)$.

4.4 Heuristics for the Splittable Hose-Model

As discussed above finding HVC embeddings is strongly NP-hard in the unsplittable path model. Furthermore, using Mixed-Integer Program 1 to obtain (optimal) solutions in *reasonable* time seems out of reach.

To compute hose-based embeddings more efficiently, we have to drop (1) the flexible node mapping as well as (2) the unsplittable path routing model. Accordingly, this section presents the polynomial-time Linear Program 2 for computing optimal Hose Multi-Path Routings (HMPR) under fixed node mappings. In contrast to HVC-OSPE, the formulation HMPR can be solved within few minutes for much larger problems.[2] Together with an efficient node mapping heuristic based on algorithm VC-ACE, we obtain the effective stand-alone heuristic HVC-ACE for the (splittable) hose-based virtual cluster embedding.

Based on the Mixed-Integer Program 1 we obtain the *Linear Program* 2. First, if all VMs are feasibly mapped, i.e. without violating node capacities, via the function $\text{MAP}_V : V_C \to V_S$, Constraints 4-6 are not needed. However, more importantly, the computation of correct link loads can be significantly shortened, by actually discarding the $\Theta(|E_S| \cdot \mathcal{N}^2)$ many flow variables. Dropping the constraint of unsplittable path routing, Constraints 8 and 10 can be equivalently stated using the cut-set inequalities expressed in Constraint 14 (cf. Altin et al. [2]): Given any substrate node set W, to which VM i but not VM j has been mapped, there must exist a 'path' leaving W with value 1 such that the respective dual variables are bounded accordingly.

The *exponential* number of constraints can be *separated* efficiently using maximum-flow computations (cf. [11]) allowing to solve the formulation in polynomial-time [8].

[2]In our computational evaluation (see Section 5) with hundreds of nodes and edges and up to 30 node requests, HMPR computed more than 95% of the solutions in less than 3 minutes. Furthermore, our prototypical implementation of the formulation HMPR uses the simple Edmonds-Karps maximum-flow algorithm to separate the constraints 14 of Linear Program 2 using $|E_{VC}| = \Theta(|V_{VC}|^2)$ many flow computations. The runtime can be reduced significantly by using Hao and Orlin's algorithm requiring only $|V_{VC}|$ many flow computations (see [11] for an explanation).

Algorithm 2: The HVC-ACE Embedding Algorithm

```
Input:  Substrate $S = (V_S, E_S)$, request $\text{VC}(\mathcal{N}, \mathcal{B}, \mathcal{C})$,
        cost factor $k \ge 1$
Output: Splittable HVC-Embedding $\text{MAP}_V, \text{MAP}_E$
1  $E_{S'} \leftarrow \emptyset$
2  for $e \in E_S$ do
3      $E_{S'} = E_{S'} \sqcup \{e, e'\}$
4      $\text{CAP}_{S'}(e) = \text{CAP}(e)$ and $\text{CAP}_{S'}(e') = \infty$
5      $\text{COST}_{S'}(e) = \text{COST}(e)$ and $\text{COST}_{S'}(e') = \text{COST}(e) \cdot k$
6  $\text{MAP}_V, \text{MAP}_E \leftarrow$ VC-ACE$(V_S, E_{S'}, \text{VC}(\mathcal{N}, \mathcal{B}, \mathcal{C}))$
7  if $\text{MAP}_V \neq$ null then
8      $\text{MAP}_E \leftarrow$ HMPR$(\text{VC}(\mathcal{N}, \mathcal{B}, \mathcal{C}), \text{MAP}_V)$
9      if $\text{MAP}_E \neq$ null then
10         return $\text{MAP}_V, \text{MAP}_E$
11 return null
```

Lastly, the splittable routing for $(i,j) \in E_{VC}$ can be recovered from the dual variables by computing a minimum-cost flow of value 1 from $\text{MAP}_V(i)$ to $\text{MAP}_V(j)$ with edge capacities $\omega^i_{uv} + \omega^j_{uv}$ for $(u,v) \in E_S$. Such a flow always exists as Constraint 14 enforces that any cut must have at least capacity 1 (cf. Constraint 10 of MIP 1).

The Linear Program 2 therefore allows us to compute optimal splittable hose-routings in polynomial-time for *fixed* node mappings. In the following we describe a heuristic to find a corresponding node mapping to obtain the embedding algorithm HVC-ACE. Since optimal VC and HVC solutions coincide in uncapacitated networks (see Section 4.2), we employ an adaption of the VC mapping algorithm VC-ACE (see Algorithm 1). For each edge in the original graph, a copy of infinite capacity is added, having k times the original cost. Clearly, on this adapted graph $(V_S, E_{S'})$ a feasible solution – given the existence of a feasible node mapping – always exists. By varying the factor k, the cost of using an infinite-capacity edge can be controlled. Setting $k = 1$, the cost structure of the original graph is not changed, while by setting $k = \infty$ the number of non-existent edges can be minimized. Thus, for $k = \infty$, the node mapping equals the one found by algorithm VC-ACE, if one existed.

5. EVALUATION

This section compares the performance of the optimal VC embedding and our heuristic hose-based embeddings. In particular we show that by using Algorithm HVC-ACE, the chances of being able to accept a single request can be up to 60% higher than in the classic VC model. We furthermore show that the hose-abstraction may reduce the resource footprint by up to 25% on fat tree topologies.

Our evaluation setup is as follows. We consider the fat tree (12-port switches, 432 servers) and the MDCube (4 BCubes with $n = 12$ and $k = 1$) datacenter topologies. We assume uniform edge capacities and each server offers 2 VM slots. Requests are embedded over time in an online fashion with exponentially distributed inter-arrival times and duration. The mean of the inter-arrival time is chosen to impose a system load of 75% (w.r.t. node utilization if all requests can be accepted). The size $\mathcal{N}$ of the virtual clusters is chosen uniformly at random from the set $\{10, \ldots, 30\}$. $\mathcal{B}$ is chosen uniformly at random between 20% to 100% of the available bandwidth of a single (unused) substrate link. To impose an initial system load, requests are embedded within

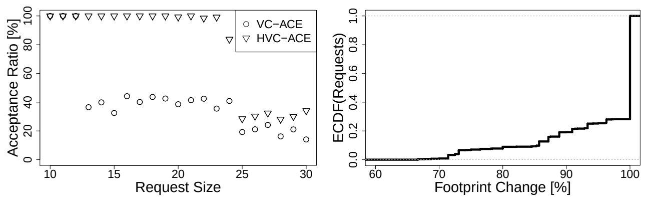

Figure 4: Experiments on a 432 server fat tree. Acceptance ratio of VC-ACE and HVC-ACE *(left)* and footprint benefits of HVC-ACE compared to VC-ACE *(right)*.

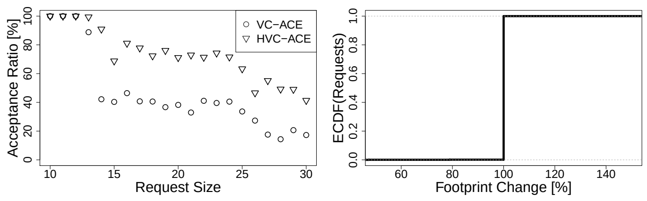

Figure 5: Experiments on a 576 server MDCube. Acceptance ratio of VC-ACE and HVC-ACE *(left)* and footprint benefits of HVC-ACE compared to VC-ACE *(right)*.

the first 45 time units (three generations of requests) using VC-ACE if possible. Then, a single data point is generated by embedding the next given request, using VC-ACE and HVC-ACE. Here HVC-ACE denotes the best solution found for the cost parameter $k \in \{1, 5, 10, \infty\}$.

We evaluate two metrics: the acceptance ratio and the relative resource footprint. The *acceptance ratio* captures the ratio of requests that can be successfully embedded. The *relative* resource footprint is the quotient of the embedding costs of the solutions of HVC-ACE and VC-ACE, if VC-ACE found a solution. As we consider unit node and edge costs, the cost is proportional to the edge usage.

Figure 4 *(left)* shows a significant advantage of HVC-ACE in terms of acceptance ratio: starting from 13 nodes, VC-ACE can only embed roughly 40% of the requests. At 25 nodes, the acceptance ratio drops, to roughly 20%. The acceptance ratio of HVC-ACE remains close to 100% for up to 23 nodes. The drops of the acceptance ratio are related to the number of ports of the switches (12 port switches in the fat tree). Figure 4 *(right)* shows the impact of using HVC-ACE instead of VC-ACE on the footprint (when VC-ACE found a solution). The x-axis plots the relative footprint of HVC-ACE normalized by the footprint of VC-ACE. By adopting the hose model (i.e. when using HVC-ACE) the footprint can be reduced for roughly a quarter of all requests by up to 30%. Note that for 20% of the requests, the footprint was reduced by at least 10%.

The advantages of the HVC interpretation on hypercubic topologies are depicted in Figure 5: although HVC-ACE does not yield lower footprints than VC-ACE (if a solution was found), HVC-ACE can still provide an average improvement of 30% in the acceptance ratio, for $\mathcal{N} \geq 14$.

6. CONCLUSION

In this paper, we revisited the virtual cluster embedding problem which has recently been studied intensively on fat trees. We showed that the problem is not NP-hard, but can be solved in polynomial-time on arbitrary substrate topologies using our algorithm VC-ACE. We have introduced the hose-based interpretation of virtual clusters and proposed the efficient embedding algorithm HVC-ACE. Our evaluation shows that the hose abstraction can significantly improve the acceptance ratio by up to 60% and may yield better solutions in terms of resource utilization on fat trees.

Acknowledgement. This research was supported by the EU projects UNIFY FP7-IP-619609 and Bigfoot FP7-ICT-317858, as well as by the German BMBF Software Campus grant 01IS12056. The authors would like to thank the anonymous reviewers for their valuable feedback.

7. REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable, commodity data center network architecture. In *Proc. ACM SIGCOMM*, pages 63–74, 2008.

[2] A. Altin, E. Amaldi, P. Belotti, and M. c. Pinar. Provisioning virtual private networks under traffic uncertainty. *Networks*, 49(1):100–115, Jan. 2007.

[3] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards predictable datacenter networks. In *Proc. ACM SIGCOMM*, 2011.

[4] K. Chowdhury, M. R. Rahman, and R. Boutaba. Virtual network embedding with coordinated node and link mapping. In *Proc. INFOCOM 2009 and IEEE/ACM Trans. Netw. (ToN) 2012*, 2012.

[5] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica. Managing Data Transfers in Computer Clusters with Orchestra. In *ACM SIGCOMM*, 2011.

[6] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. van der Merwe. A flexible model for resource management in virtual private networks. In *Proc. ACM SIGCOMM*, 1999.

[7] N. Goyal, N. Olver, and F. B. Shepherd. The VPN conjecture is true. *Proc. ACM STOC*, 2008.

[8] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.

[9] 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. In *Proc. ACM SIGCOMM*, pages 63–74, 2009.

[10] A. Gupta, J. Kleinberg, A. Kumar, R. Rastogi, and B. Yener. Provisioning a virtual private network. In *Proc. ACM STOC*, 2001.

[11] T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32(3):207–232, 1998.

[12] B. Korte and J. Vygen. *Combinatorial optimization*. Springer, 2002.

[13] J. C. Mogul and L. Popa. What we talk about when we talk about cloud network performance. *SIGCOMM CCR*, 2012.

[14] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey. Jellyfish: Networking data centers randomly. In *Proc. USENIX NSDI*, 2012.

[15] H. Wu, G. Lu, D. Li, C. Guo, and Y. Zhang. Mdcube: a high performance network structure for modular data center interconnection. In *Proc. ACM CoNEXT*, 2009.

[16] D. Xie, N. Ding, Y. C. Hu, and R. Kompella. The only constant is change: Incorporating time-varying network reservations in data centers. In *Proc. ACM SIGCOMM*, pages 199–210, 2012.

[17] Measuring EC2 system performance. `http://goo.gl/V5zhEd`.