

Slice Embedding Solutions for Distributed Service Architectures

Flavio Esposito Ibrahim Matta Vatche Ishakian
 flavio@cs.bu.edu matta@cs.bu.edu visahak@cs.bu.edu

Computer Science Department
 Boston University
 Boston, MA

Technical Report BUCS-TR-2011-025

Abstract—Network virtualization provides a novel approach to run multiple concurrent virtual networks over a common physical network infrastructure. From a research perspective, this enables the networking community to concurrently experiment with new Internet architectures and protocols. From a market perspective, on the other hand, this paradigm is appealing as it enables infrastructure service providers to experiment with new business models that range from leasing virtual slices of their infrastructure to host multiple concurrent network services.

In this paper, we present the slice embedding problem and recent developments in the area. A slice is a set of virtual instances spanning a set of physical resources. The embedding problem consists of three main tasks: (1) resource discovery, which involves monitoring the state of the physical resources, (2) virtual network mapping, which involves matching users' requests with the available resources, and (3) allocation, which involves assigning the resources that match the users' query.

We also outline how these three tasks are tightly connected, and how there exists a wide spectrum of solutions that either solve a particular task, or jointly solve multiple tasks along with the interactions among them. To dissect the space of solutions, we introduce three main classification criteria, namely, (1) the type of constraints imposed by the user, (2) the type of dynamics considered in the embedding process, and (3) the allocation strategy adopted. Finally, we conclude with a few interesting research directions.

I. INTRODUCTION

We all became familiar with the layered reference model of ISO OSI as well as the layered TCP/IP architecture [47]. In these models, a layer is said to provide a *service* to the layer immediately above it. For example, the transport layer provides services (logical end-to-end channels) to the application layer, and the internetworking layer provides services (packet delivery across individual networks) to the transport layer.

The notion of distributed service architecture extends this service paradigm to many other (large scale) distributed systems.

Aside from the Internet itself, including its future architecture design, *e.g.*, NetServ [73] or RINA [23], with the term *distributed service architecture* we refer to a large scale distributed system whose architecture is based on a service paradigm.

Some examples are datacenter-based systems [39], Cloud Computing [36] (including high performance computing systems such as cluster-on-demand services), where the rentable resources can scale both up and down as needed, Grid Computing [45], overlay networks (*e.g.*, content delivery networks [6],

[10]), large scale distributed testbed platforms (*e.g.*, PlanetLab [65], Emulab/Netbed [77], VINI [7], GENI [31]), or Service-oriented Architecture (SoA), where web applications are the result of the composition of services that need to be instantiated across a collection of distributed resources [80].

A common characteristic of all the above distributed systems is that they all provide a service to a set of users or, recursively, to another service. In this survey, we restrict our focus on a particular type of service: a slice. We define a slice to be a set of virtual instances spanning a set of physical resources.

The lifetime span of a slice ranges from few seconds (in the case of cluster-on-demand services) to several years (in case of a virtual network hosting a content distribution service similar to Akamai, or even a GENI experiment hosting a novel architecture looking for new adopters to opt-in [34]). Therefore, the methods to acquire, configure, manipulate and manage such slices could be different across different service architectures. In particular, the problem of discovering, mapping and allocating physical resources (slice embedding) has different time constraints in each service architecture.¹

In some distributed service architecture applications, *e.g.* virtual network testbed, the slice creation and embedding time is negligible relative to the running time of the service they are providing. In many other applications, *e.g.* financial modeling, anomaly analysis, or heavy image processing, the time to solution — instant between the user, application or service requests a slice and the time of task completion — is dominated by or highly dependent on the slice creation and embedding time.

Therefore, to be profitable, most of those service architectures require agility—the ability to allocate and deallocate any physical resource (node or link) to any service at any time². Those stringent requirements, combined with the imperfect design of today's data center networks [35] and with the lack of an ideal virtualization technology [78], have recently re-motivated research on resource allocation [13], [82], [51], [35], [4], [70].

In this paper, we define the slice embedding problem—a

¹By resources we mean processes, storage capacity, and physical links, as well as computational resources such as processors.

²We extend the definition of agility as “ability to assign any server to any service” given by Greenberg *et al.* [35] by including links and, other resources along with a deallocation phase.

subarea of the resource allocation for service architectures—in Section II, we give a taxonomy (Section III), and we survey some of the recent solutions for each of its tasks (Sections IV, V and VI). Then, with the help of optimization theory, we model the three phases of the slice embedding problem as well as its tasks’ interactions (Section VIII). We point out how all the proposed approaches—including the related facility location problems (Section VII)—have considered either cases where the time to solution is practically equivalent to the running time of a slice, *i.e.* they did not consider the slice creation and embedding time at all, or they did not model some of the slice embedding tasks. In Section IX we discuss some interesting open research directions and finally, in Section X we conclude our discussion.

II. BACKGROUND AND AREA DEFINITION

A. Network Virtualization

Network virtualization provides a novel approach to running multiple concurrent virtual networks over a common physical network infrastructure. A physical network supports virtualization if it allows the coexistence of multiple virtual networks. Each virtual network is a collection of virtual nodes and virtual links that connect a subset of the underlying physical network resources. The most important characteristic of such virtual networks is that they are customizable (*i.e.*, can concurrently run different protocols or architectures, each tailored to a particular service or application [75]).

The interest in this technology has recently grown significantly because it will help the research community in the testing of novel protocols and algorithms in pseudo-real network environments [65], [77], [7], [28], as well as experimenting with novel Internet architectures as envisioned in [3]. This paradigm is particularly appealing to providers as it enables new business models: operators may in fact benefit from diversifying their infrastructure by leasing virtual networks to a set of customers [30], or by sharing costs in deploying a common infrastructure [11].

A recent survey on network virtualization can be found in [18]. The authors compare with a broad perspective, approaches related to network virtualization, *e.g.* virtual private networks and overlay networks. The paper also discusses economic aspects of service providers, analyzes their design goals (such as manageability or scalability), and overviews recent projects that use this technology (*e.g.* Planetlab [65] and GENI [31]). We narrow our focus on a more specific subarea of network virtualization (*i.e.* slice embedding), introducing a new taxonomy inspired by optimization theory for the three phases of the slice embedding problem. We leave our utility functions and model constraints as general as possible, so they can be instantiated, refined or augmented based on policies that would lead to efficient slice embedding solutions.

B. The Slice Embedding Problem

In this paper, we focus on a particular aspect of network virtualization, namely, the slice embedding problem.

A slice is defined as a set of virtual instances spanning a set of physical resources of the network infrastructure. The

slice embedding problem comprises the following three steps: resource discovery, virtual network mapping, and allocation.

Resource discovery is the process of monitoring the state of the substrate (physical) resources using sensors and other measurement processes. The monitored states include processor loads, memory usage, network performance data, etc. We discuss the resource discovery problem in Section IV.

Virtual network mapping is the step that matches users’ requests with the available resources, and selects some subset of the resources that can potentially host the slice. Due to the combination of node and link constraints, this is by far the most complex step in the slice embedding problem. In fact this problem is NP-hard [19]. These constraints include intra-node (*e.g.*, desired physical location, processor speed, storage capacity, type of network connectivity), as well as inter-node constraints (*e.g.*, network topology). We define the virtual network mapping problem in Section V.

Allocation involves assigning the resources that match the user’s query to the appropriate slice. The allocation step can be a single shot process, or it can be repeated periodically to either reassign or to acquire additional resources for a slice that has already been embedded.

C. Interactions in the Slice Embedding Problem

Before presenting existing solutions to the tasks encompassing the slice embedding problem, it is important to highlight the existence of interactions among these tasks, the nature of these interactions, how they impact performance, as well as the open issues in addressing these interactions.

In Figure 1, a user is requesting a set of resources. The arrow (1) going from the “Requests” to the “Discovery” block, represents user queries that could potentially have multiple levels of expressiveness and a variety of constraints. The resource discoverer (2) returns a subset of the available resources (3) to the principle in charge of running the virtual network mapping algorithm (4). Subsequently, the slice embedding proceeds with the allocation task. A list of candidate mappings (5) are passed to the allocator (6), that decides which physical resources are going to be assigned to each user. The allocator then communicates the list of winners (7)—users that won the allocation—to the discoverer, so that future discovery operations can take into account resources that have already been allocated. It is important to note that the slice embedding problem is essentially a closed feedback system, where the three tasks are solved repeatedly—the solution in any given iteration affects the space of feasible solutions in the next iteration.

D. Solutions to the Slice Embedding Problem

Solutions in the current literature either solve a specific task of the slice embedding problem, or are hybrids of two tasks. Some solutions jointly consider resource discovery and network mapping [41], [1], others only focus on the mapping phase [81], [54], [21], or on the interaction between virtual network mapping and allocation [79], [52], while others consider solely the allocation step [5], [9], [49], [33], [20]. Moreover, there are solutions that assume the virtual network mapping

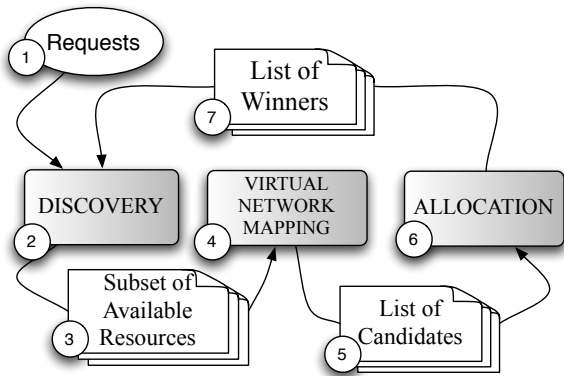


Fig. 1. Interactions and data exchanges in the slice embedding problem.

task is solved, and only consider the interaction between the resource discovery and allocation [68]. We do not discuss solutions that address the resource discovery task in isolation, since it is not different from classical resource discovery in the distributed system literature (see [60] for an excellent survey on the topic). In addition to considering one [81], [5] or more [62], [79] tasks, solutions also depend on whether their objective is to maximize users' or the providers' utility.

E. The novelty of the slice embedding problem

The slice embedding problem, or more specifically its constituent tasks, and network virtualization in general, may seem identical to problems in classical distributed systems. Network virtualization, however, is different in several ways, namely: (a) it enables novel business models, (b) it enables novel coexisting network approaches, and (c) it creates new embedding challenges that must be addressed.

Business models: network virtualization lays the foundations for new business models [22]. Network resources are now considered commodities to be leased on demand. The leaser could be an infrastructure or service provider, and the lessee could be another service provider, an enterprise, or a single user (e.g. a researcher in the case of virtual network testbed as in [31], [7], [38], [65], [28]). In those cases where the infrastructure is a public virtualizable network testbed (e.g. GENI [31]), the physical resources may not have any significant market value, since they are made available at almost no cost to research institutions.

Coexisting network approaches: the concept of multiple coexisting logical networks appeared in the networking literature several times in the past. The most closely related attempts are virtual private networks and overlay networks. A virtual private network (VPN) is a dedicated network connecting multiple sites using private and secured tunnels over a shared communication network. Most of the time, VPNs are used to connect geographically distributed sites of a single enterprise: each VPN site contains one or more customer edge devices attached to one or more provider edge routers [66].

An overlay network, on the other hand, is a logical network built on top of one or more existing physical networks. One substantial difference between overlays and network virtualization is that overlays in the existing Internet are typically implemented at the application layer, and therefore they may have limited applicability.

For example, they falter as a deployment path for radical architectural innovations in at least two ways: first, overlays have largely been in use as means to deploy narrow fixes to specific problems without any holistic view; second, most overlays have been designed in the application layer on top of the IP protocol, hence, they cannot go beyond the inherent limitations of the existing Internet [3].

In the case of VPNs, the virtualization level is limited to the physical network layer while in the case of overlays, virtualization is limited to the end hosts. Network virtualization introduces the ability to access, manage and control each layer of the current Internet architecture in the end hosts, as well as providing dedicated virtual networks.

Embedding challenges: although the research community has explored the embedding of VPNs in a shared provider topology, e.g., [26], usually VPNs have standard topologies, such as a full mesh. A virtual network in the slice embedding problem, however, may represent any topology. Moreover, resource constraints in a VPN or overlays are limited to either bandwidth requirements or node constraints, while in network virtualization, both link and node constraints may need to be present simultaneously. Thus, the slice embedding problem differs from the standard VPN embedding because it must deal with both node and link constraints for arbitrary topologies.

III. TAXONOMY

To dissect the space of existing solutions spanning the slice embedding tasks, as well as interactions among them, we consider three dimensions as shown in Figure 2: the *type of constraint*, the *type of dynamics*, and the *resource allocation approach*.

A. Constraint type

Users need to express their queries efficiently. Some constraints are on the nodes and/or links (e.g., minimum CPU requirement, average bandwidth, maximum allowed latency) while others consider inter-group [1] or geo-location constraints [17].

Based on this dimension, research work in this area assumes no constraints [81], considers constraints on nodes only [65], links only [55], [67], [37], or on both nodes and links [5], [79]. In addition, the order in which the constraints are satisfied is important as pointed out in [52]: satisfy the node constraints and then the link constraints [81], [79], or satisfy both constraints simultaneously [54], [52].

B. Dynamics

Each task in the slice embedding problem may differ in terms of its dynamics. In the resource discovery task, the

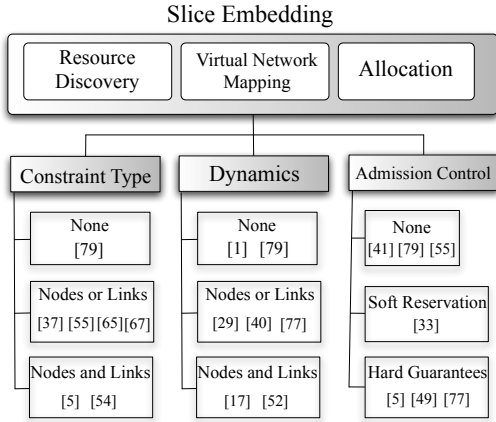


Fig. 2. Overview of the slice embedding taxonomy with classification of representative references.

status updates of each physical resource may be collected periodically [41], or on demand [1].

In the virtual network mapping task, virtual resources may be statically mapped to each physical resource [81], or they can move (*e.g.*, using path migrations [79] or by re-running the mapping algorithm [29]) to maximize some notion of utility [37]. Also, the mapping can focus only on one single phase at a time where each phase considers only nodes or links [81], [40], or simultaneously both nodes and links [52], [17].

Finally, the allocation task may be dynamic as well: users may be swapped in or out to achieve some Quality of Service (QoS) or Service Level Agreement (SLA) performance guarantees, or they can statically remain assigned to the same slice. An example of static assignment of a slice may be an infrastructure hosting a content distribution service similar to Akamai, whereas an example of dynamic reallocation could be a researcher’s experiment being swapped out from/into the Emulab testbed [77].

C. Admission Control

As the substrate—physical infrastructure—resources are limited, some requests must be rejected or postponed to avoid violating the resource guarantees for existing virtual networks, or to maximize profit of the leased network resources. Some research work, however, does not consider any resource allocation [41], [54], [21], [81], [55], [52]. Others consider the resource allocation task, with [33] or without [49], [5], [79] guarantees to the user, *i.e.*, the resource allocation mechanism enforces admission to the users, or it only implements a tentative admission, respectively. An example of tentative admission is a system that issues tickets, without guarantee that those tickets can be exchanged with a resource later in time. The literature defines those tentative admission mechanisms that do not provide hard guarantees as *soft reservation* [33].

IV. RESOURCE DISCOVERY

Although researchers have developed, and in some cases deployed a number of resource discovery solutions for wide-

area distributed systems, the research in this area still has many open problems. Some of the existing distributed systems provide resource discovery through a centralized architecture, see, *e.g.*, Condor [53], Assign [67], or Network Sensitive Service Discovery (NSSD) [41]; others use a hierarchical architecture such as Ganglia [58], while XenoSearch [72], SWORD [62] and iPlane Nano [57] employ a decentralized architecture.

All of these systems allow users to find nodes that meet per-node constraints, except iPlane Nano that considers path metrics, while NSSD, SWORD, and Assign also consider network topologies. Unfortunately, none of these solutions analyze the resource discovery problem when the queried resources belong to multiple infrastructure or service providers. To obtain an efficient slice embedding, such cases would in fact require some level of cooperation (*e.g.*, by sharing some state), and such incentives to cooperate may be scarce.

As mentioned previously, we do not discuss solutions that address the resource discovery task in isolation, since it is not different from classical resource discovery in the distributed systems literature. Instead, we consider the resource discovery problem in combination with either the allocation or the network mapping task.

A. Discovery + allocation

We first discuss the interaction between discovery and allocation described in Network Sensitive Service Discovery (NSSD) [41]. The goal is to discover a service that meets a set of network properties specified by the user, and allocate it to the user.

This work emphasizes the importance of the interaction between discovery of network resources and their allocation to the users. The resource discovery task infers the network’s performance metrics during its search and returns the best match with respect to some user criteria. In general, once a user’s query is received, in existing systems either the provider (pure provider-side allocation) or the users (pure user-side allocation) execute the allocation task. If the allocation is done by the provider, users do not have to worry about anything after they submit a query, but may not know the quality of service they are going to get (in systems like PlanetLab for example, there are no service level agreements that the provider needs to meet). On the other hand, when the allocation is done by the user, each user needs to obtain a long list of candidates, as well as collect the status information of each candidate. Thus, the overhead of the discovery task is higher if users need to have the ability to choose the best set of resources. When the provider does the allocation instead, there may be no need to look at the complete set of resources as some heuristic (*e.g.* first fit) can be applied. Moreover, by showing the most available physical resources they own, providers could (indirectly) have to release information about their states, *e.g.*, information about which customer is hosted on a physical machine could be inferred [69].

To the best of our knowledge, NSSD is the first system that integrates the discovery and allocation tasks while enabling users to query static and dynamic network properties. Compared with pure provider-side allocation, NSSD allows users to

control the selection criteria by returning a list of candidates. Compared with pure user-side allocation, NSSD has lower overhead in the discovery task, as only a small number of candidates are returned. In this work, the resources to allocate are single servers, hence there is no virtual network mapping phase.

B. Discovery + virtual network mapping

We present SWORD [1], a system that considers the interaction between the resource discovery and the virtual network mapping tasks. SWORD is a resource discovery infrastructure for shared wide-area platforms such as PlanetLab [65]. We choose to describe SWORD as it is a well known network discovery system whose source code is available [74]. The system has been running on PlanetLab for several years. Some of the functionalities described in the original paper, however, are currently disabled. For example, the current implementation of SWORD runs in centralized mode, and inter-node and group requirements (*i.e.*, constraints on links and set of nodes, respectively), are not supported because no latency or bandwidth estimates are available.

Users wishing to find nodes for their application submit a resource request expressed as a topology of interconnected groups. A group is an equivalence class of nodes with the same per-node requirements (*e.g.*, free physical memory) and the same inter-node requirements (*e.g.*, inter-node latency) that is within each group. Supported topological constraints within and among groups include the required bandwidth and latency.

In addition to specifying absolute requirements, users can supply SWORD with per-attribute *penalty functions*, that map the value of an attribute (feature of a resource, such as load or delay) within the required range but outside an ideal range, to an abstract penalty value. This capability allows SWORD to rank the quality of the configurations that meet the applications' requirements, according to the relative importance of each attribute. Notice that these penalty values would be passed to the allocation together with the list of candidates.

Architecturally, SWORD consists of a distributed query processor and an *optimizer* which can be viewed as a virtual network mapper. The distributed query processor uses multi-attribute range search built on top of a peer-to-peer network to retrieve the names and attribute values of the nodes that meet the requirements specified in the user's query. SWORD's optimizer then attempts to find the lowest-penalty assignment of platform nodes (that were retrieved by the distributed query processor) to groups in the user's query—that is, the lowest-penalty embedding of the requested topology in the PlanetLab node topology, where the penalty of an embedding is defined as the sum of the per-node, inter-node, and inter-group penalties associated with that selection of nodes.

Due to the interaction between the distributed query processor (resource discovery task) and the optimizer (mapping task), SWORD is more than a pure resource discoverer. SWORD provides resource discovery, solves the network mapping task, but does not provide resource allocation. In particular, since PlanetLab does not currently support resource guarantees, a set of resources that SWORD returns to a user may no longer

meet the resource request at some future point in time. In light of this fact, SWORD supports a *continuous query* mechanism where a user's resource request is continually re-matched to the characteristics of the available resources, and in turn a new set of nodes are returned to the user. The user can then choose to migrate one or more instances of their application. This process is all part of the general feedback system outlined in Figure 1.

V. VIRTUAL NETWORK MAPPING

The virtual network mapping is the central phase of the slice embedding problem. In this section we define the problem of virtual network mapping, then we survey solutions that focus only on this phase, as well as solutions that cover interactions with the other two tasks of the slice embedding problem.

A. Problem definition

The virtual network mapping problem is defined as follows [52]:

Definition 1 (Network): A Network is defined as an undirected graph $G = (N, L, C)$ where N is a set of nodes, L is a set of links, and each node or link $e \in N \cup L$ is associated with a set of constraints $C(e) = \{C_1(e), \dots, C_m(e)\}$. A physical network will be denoted as $G^P = (N^P, L^P, C^P)$, while a virtual network will be denoted as $G^V = (N^V, L^V, C^V)$.

Definition 2 (Virtual Network Mapping): Given a virtual network $G^V = (N^V, L^V, C^V)$ and a physical network $G^P = (N^P, L^P, C^P)$, a virtual network mapping is a mapping of G^V to a subset of G^P , such that each virtual node is mapped onto exactly one physical node, and each virtual link is mapped onto a loop-free path p in the physical network. The mapping is called valid if all the constraints $C(e)$ of the virtual network are satisfied and do not violate the constraints of the physical network. More formally, the mapping is a function

$$M : G^V \rightarrow (N^P, \mathcal{P}) \quad (1)$$

where \mathcal{P} denotes the set of all loop-free paths in G^P . M is called a *valid mapping* if all constraints³ of G^V are satisfied, and for each $l^v = (s^V, t^V) \in L^V$, \exists a path $p : (s^P, \dots, t^P) \in \mathcal{P}$ where s^V is mapped to s^P and t^V is mapped to t^P .

Due to the combination of node and link constraints, the virtual network mapping problem is NP-hard. For example, assigning virtual nodes to the substrate (physical) network without violating link bandwidth constraints can be reduced to the multiway separator problem which is NP-hard [2].

To reduce the overall complexity, several heuristics were introduced, including backtracking algorithms [54], [52], simulated annealing as in Emulab [67], as well as heuristics that solve the node and link mapping independently.

³Examples of node constraints include CPU, memory, physical location, whereas link constraints may be delay, jitter, or bandwidth.

TABLE OF NOTATIONS		
Symbol	Page	Meaning
G	6	Undirected graph representing a general network
N	6	General set of nodes (or vertices) of a network
L	6	General set of links (or edges) of a network
C	6	General set of network constraints
C^P (C^V)	6	General set of physical (virtual) network constraints
$C(e) = \{C_1(e), \dots, C_m(e)\}$	6	Set of m constraints on the element e (node or link) of the network
G^P (G^V)	6	Undirected graph representing a physical (virtual) network
N^P (N^V)	6	Set of nodes or vertices of a physical (virtual) network
L^P (L^V)	6	Set of links or edges of a physical (virtual) network
\mathcal{P}	6	Set of loop-free physical paths in a physical network G^P
$l^v = (s^V, t^V)$	6	Virtual link starting from virtual node s^V , and ending in virtual node t^V
$p : (s^P, \dots, t^P) \in \mathcal{P}$	6	Physical path starting from physical node s^P , and ending in physical node t^P
M	6	Mapping function: $G^V \rightarrow (N^P, \mathcal{P})$
u'	7	Next physical node assigned in node mapping algorithm [81]
S_{max} (S_{lmax})	7	Maximum node (link) stress in G^P [81]
$S_N(v)$ ($S_L(l)$)	7	Current node (link) stress in G^P [81]
l	7	Index of physical links [81]
v	7	Index of physical nodes to map [81]
u	7	index of mapped physical nodes in node mapping algorithm [81]
$L(v)$	7	Set of links adjacent to physical node v [81]
$D(v, u)$	7	Distance between physical node v and u [81]
$\Pi(G^V)$	7	Revenue for allocating virtual network G^V [79]
CPU_r and bw_r	7	CPU and bandwidth required by the virtual network [79]
CPU_a and bw_a	7	CPU and bandwidth available on a physical network [79]
Ω	7	Price normalization factor [79]
$H(n^P)$	7	available resource on physical node n^P [81]
R_N (R_L)	8	Physical node (link) stress ratio [79]
$U^k(\cdot)$	8	Convex objective function run by virtual network k [37]
n_0	8	Number of virtual networks to simultaneously map [37]
$C^{(k)} = c_{lj}^{(k)}$	8	Binary matrix of capacity constraints for virtual network k using virtual path j on physical link l [37]
$y^{(k)}$	8	virtual link capacities for virtual network k [37]
$z^{(k)}$	8	Path rate vector for virtual network k [37]
$g^{(k)}$	8	General convex constraint for virtual network k [37]
\mathcal{D}	8	Matrix of physical link capacity
$w^{(k)}$	8	Weight assigned to virtual network k in the slice allocation phase
ω_{ij}	9	Weight (or utilization) imposed on resource j by user i ,
P_j	9	Price (in dollars) of the resource j [43]
U_j	9	Overall utilization of resource j [43]
\mathcal{R}_j	9	Physical CPU capacity of resource j in a <i>Colocation Game</i> [43]
$\mathcal{K}_j(i)$	9	Colocation cost for user i when mapped to resource j
a_{ij}	10	binary variable representing element i in the j^{th} set in a Set Packing Problem
w_j	10	Weight assigned to user requesting the set of resources—or objects— j in any allocation (Set Packing Problem)
y_j	10	Binary allocation variable for object j in a Set Packing Problem
$W(O)$	10	Set of users W (objects O) to be allocated in a Set Packing Problem
Q	10	Collection of subsets of objects in a Set Packing Problem
b_i	10	Number of copies for each object i in a Set Packing Problem
c_i	11	Cost of opening a facility at location i in a Facility Location Problem
d_{ij}	11	Cost of serving a user j from facility i
z_i	11	Binary variable showing whether or not the facility is selected at location i
x_{ij}	11	Binary variable that associates user j served by facility i in Facility Location Problem
x_i	11	Decision variable for location i , which is equal to one if the facility is selected
$f(\cdot), g(\cdot), h(\cdot)$	12	Utility functions for the discovery, virtual network mapping and allocation phase
γ (γ_j)	12	Number of virtual nodes (requested by user j)
ψ (ψ_j)	12	Number of virtual links (requested by user j)
n_{ij}^V (n_{ij}^P)	12	Decision variable on virtual (physical) node mappable (mapped) to user j
l_{ij}^V (p_{ij})	12	Decision variable on virtual (physical loop-free path) link mappable (mapped) to user j
Θ_{ij} (Φ_{kj})	12	System's revenue when user j gets assigned to virtual node i (virtual link k .)
C_i^n (C_k^l)	12	Max virtual nodes (links) that can be simultaneously hosted on the physical node i (physical path k)

TABLE I
NOTATIONS USED IN THE PAPER.

B. Network mapping without constraints

The problem of static assignments of resources to a virtual network has been investigated in [81]. Since it is NP-hard, the authors proposed a heuristic to select physical nodes with lower *stress* (*i.e.*, with the lower number of virtual nodes already assigned to a given physical node), in an attempt to balance the load. The algorithm consists of two separate phases: node mapping and link mapping. The node mapping phase consists of an initialization step—cluster center localization—and an iterative subroutine—substrate node selection—that progressively selects the next physical node u' to which the next virtual node is mapped, *i.e.* the physical node with the least stress.

In particular, the center cluster is selected as follows:

$$u' = \arg \max_v \left\{ [S_{nmax} - S_N(v)] \sum_{l \in L(v)} [S_{lmax} - S_L(l)] \right\}$$

where S_{nmax} and S_{lmax} are the maximum node and link stress seen so far in the physical network, respectively. $S_N(v)$ is the stress on the physical node v , while $S_L(l)$ is the stress on the physical link l . $[S_{nmax} - S_N(v)]$ captures the availability of node v , while the availability on the links adjacent to v is captured by $\sum_{l \in L(v)} [S_{lmax} - S_L(l)]$.

The substrate node selection subroutine maps the remaining virtual nodes by minimizing a potential function proportional to both node and link stress on the physical network, *i.e.*:

$$u' = \arg \min_v \frac{\sum_{u \in V_A} D(v, u)}{S_{nmax} - S_N(v) + \epsilon}$$

where V_A is the set of already selected substrate nodes, v is an index over all physical nodes (so v could be the same as some u), ϵ is a small constant to avoid division by zero, and D is the distance between any two physical nodes v and u and it is defined as:

$$D(v, u) = \min_{p \in \mathcal{P}(u, v)} \sum_{l \in p} \frac{1}{S_{lmax} - S_L(l) + \epsilon}$$

where p is an element of all loop-free paths $\mathcal{P}(u, v)$ on the physical network that connects nodes u and v . The node mapping phase successfully terminates when all the virtual nodes are mapped.

The link mapping invokes a shortest path algorithm to find a minimum hop (loop-free) physical path connecting any pair of virtual nodes.

In the same paper, the authors modify this algorithm by subdividing the complete topology of a virtual network into smaller star topologies. These sub-topologies can more readily fit into regions of low stress in the physical network.

C. Network mapping with constraints

Many of the solutions to the virtual network mapping problem consider some constraints in the query specification. Lu and Turner [55] for example, introduce flow constraints in a mapping of a single virtual network. The NP-hard mapping problem is solved by greedily finding a backbone-star topology of physical nodes (if it exists, otherwise the slice cannot be

embedded), and the choice is refined iteratively by minimizing a notion of cost associated with the candidate topologies. The cost metric of a virtual link is proportional to the product of its capacity and its physical length. No guarantees on the convergence to an optimal topology mapping are provided, and only bandwidth constraints are imposed.

A novel outlook on the virtual network mapping problem for virtual network testbeds is considered in [21]. A topology and a set of (upper and lower bound) constraints on the physical resources are given, and a feasible mapping is sought. In order to reduce the search space of the NP-hard problem, a depth-first search with pruning as soon as a mapping becomes infeasible is used.

Another solution that considers embedding with constraints is presented in [52]. The authors propose a backtracking algorithm based on a subgraph isomorphism search method [48], that maps nodes and links simultaneously. The advantage of a single step node-link approach is that link constraints are taken into account at each step of the node mapping, therefore when a bad decision is detected, it can be adjusted by backtracking to the last valid mapping. With a two-stage approach instead, the remapping would have to be done for all the nodes, which is computationally expensive.

D. Network mapping + allocation

In all the solutions that focus only on the virtual network mapping task, only a single virtual network is considered (with or without constraints), and no resource allocation mechanism is provided. In case the mapping algorithm is designed for virtual network testbeds such as Emulab [77] or Planetlab [65], this may not be an issue except in rare cases, *e.g.*, during conference deadlines (see *e.g.*, Figure 1 in [5]). The lack of resource allocation is instead detrimental to an efficient slice embedding when the system aims to embed virtual networks (slices) that are profitable to the leasing infrastructure.

We discuss the case study of [79], that adds resource allocation to the virtual network mapping task, and hence introduces cooperation between the last two tasks of the slice embedding problem. The solution proposed in [79] is targeted specifically for infrastructure providers, as the physical resources considered—bandwidth and CPU—are assumed to be rentable. The authors define a revenue function R for each requested virtual network $G^V = (N^V, L^V)$ as:

$$\Pi(G^V) = \sum_{l^V \in L^V} bw_r(l^V) + \Omega \sum_{n^V \in N^V} CPU_r(n^V), \quad (2)$$

where $bw_r(l^V)$ and $CPU_r(n^V)$ are the bandwidth and the CPU requirements for the virtual link l^V and the virtual node n^V , respectively. L^V and N^V are the sets of requested virtual links and nodes, and Ω captures the price difference that the infrastructure provider may charge for CPU and bandwidth.

The algorithm is depicted in Figure 3: after collecting a set of requests, a greedy node mapping algorithm with the objective of maximizing the (long term) revenue R is run. In particular, the algorithm consists of the following three steps:

- 1) First the requests are sorted by revenue $\Pi(G^V)$ so that the most profitable mapping is sought with highest priority.

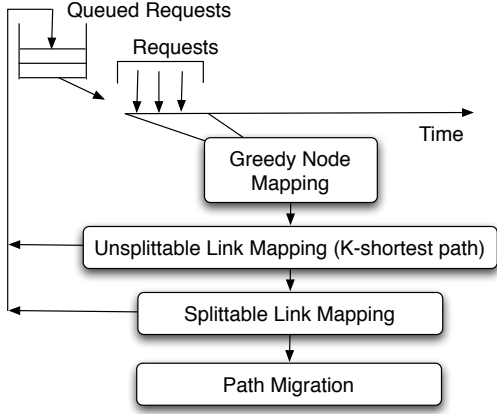


Fig. 3. Path splitting and migration mapping algorithm [79].

- 2) Then the physical nodes with insufficient available CPU capacity are discarded to reduce the complexity of the search.
- 3) Similarly to [81] (see Section V-B), a virtual node is mapped on the physical node n^P (if it exists) that maximizes the available resources H , where:

$$H(n^P) = CPU_a(n^P) \sum_{l^P \in L(n^P)} bw_a(l^P)$$

$CPU_a(n^P)$ and $bw_a(l^P)$ are the CPU and bandwidth available on the physical node n^P and link l^P , respectively, and $L(n^P)$ is the set of links adjacent to n^P .

After the node mapping, different link mapping algorithms are presented. First, the authors propose to use a *k-shortest path* algorithm [27]. The originality of this paper though, lies in the improvement of such a link assignment algorithm through two techniques: *path splitting* and *path migration*. In path splitting the virtual routers forward a fraction of the traffic through different physical paths to avoid congestion of critical physical links useful to host other virtual networks. Path migration instead is adopted to further improve the resource utilization as it consists of a periodic link mapping re-computation with a larger set of pre-mapped virtual networks, leaving unchanged both node mapping—virtual node cannot migrate on another physical node—and the path splitting ratios—fraction of the total virtual links requested to which at least two physical loop-free paths are assigned. After the link mapping algorithm, the slice requests that could not be embedded are queued for a re-allocation attempt, and they are definitively discarded if they fail a given number of attempts.

Inspired by [79] and by the PageRank algorithm [63], two topology-aware virtual network mapping and allocation algorithms (*Random Walk MaxMatch* and *Random Walk Breath First Search*) have been recently proposed [15]. The novelty, and common underlying idea of the two algorithms, is to use the same Markov chain model used in PageRank [63] to sort both physical and virtual nodes (instead of web pages), and map the most important virtual nodes to the most important physical nodes. A physical (virtual) node is highly ranked not only if it has available (required) CPU, and its adjacent links

have available (required) bandwidth (as in [79]), but also if its neighbors (recursively) have high rank.

After sorting both physical and virtual nodes, highly ranked virtual nodes are mapped to highly ranked physical nodes.

E. Dynamic approaches to network mapping and allocation

As mentioned in Section III-B, in the virtual network mapping task, virtual resources may be statically assigned to each physical resource, or they can be reassigned to maximize some notion of utility during the lifetime of a slice.

Many algorithms whose task is simply to discover feasible mappings are considered static, whether they use simulated annealing [67], genetic algorithms [77], or backtrack heuristics [54], [52]. A static resource assignment for multiple virtual networks though, especially when each virtual network needs to be customized to a particular application, can lead to lower performance and under utilization of the physical resources. Being aware of such inefficiencies, adaptive mechanisms to re-allocate physical resources, on demand or periodically, have been proposed.

Zan and Ammar [81] have proposed a dynamic version of their mapping algorithm, in which critical nodes and links in the physical network are periodically identified. To evaluate the current stress levels S_N and S_L for nodes and links, two metrics are defined: the node and link stress ratio (R_N and R_L). The former is the ratio between the maximum node stress and the average node stress across the whole physical network, while the latter is the ratio between the maximum link stress and the average link stress. Formally:

$$R_N = \frac{\max_{v \in N^P} S_N(v)}{[\sum_{v \in N^P} S_N(v)]/|N^P|}$$

$$R_L = \frac{\max_{l \in L^P} S_L(l)}{[\sum_{v \in L^P} S_L(l)]/|L^P|}$$

where N^P and L^P are the set of physical nodes and edges of the hosting infrastructure, respectively. R_N and R_L are periodically compared, and new requests are mapped optimizing the node stress if $R_N > R_L$, or the link stress if $R_N < R_L$. This process is iterated with the aim of minimizing the stress across the entire physical network.

Dynamic mapping approaches also include the solutions proposed in [55], since virtual links are iteratively reassigned, and in [79], due to the migration operations. Although without any considerations to the node constraints, also in [29] the authors consider a dynamic topology mapping for virtual networks.

A solution to the dynamic network mapping problem that uses optimization theory was presented in the *DaVinci* architecture—Dynamically Adaptive Virtual Networks for a Customized Internet [37]. A physical network with n_0 virtual mapped networks is considered. Each virtual network $k = 1, \dots, n_0$ runs a distributed protocol to maximize its own performance objective function $U^k(\cdot)$, assumed to be convex with respect to network parameters, efficiently utilizing the resources assigned to it. These objective functions, assumed to be known to a centralized authority, may vary with the

traffic class (*e.g.*, delay-sensitive traffic may wish to choose paths with low propagation-delay and keep the queues small to reduce queuing delay, while throughput-sensitive traffic may wish to maximize aggregate user utility, as a function of rate), and may depend on both virtual path rates $z^{(k)}$ and the bandwidth share $y^{(k)}$ of virtual network k over every physical link l .

The traffic-management protocols running in each virtual network are envisioned as the solution to the following optimization problem:

$$\begin{aligned} & \text{maximize} && U^{(k)}(z^{(k)}, y^{(k)}) \\ & \text{subject to} && \mathcal{C}^{(k)} z^{(k)} \leq y^{(k)} \\ & && g^{(k)}(z^{(k)}) \leq 0 \\ & && z^{(k)} \geq 0 \end{aligned} \quad (3)$$

where $z^{(k)}$ are the variables (virtual path rates), $g^{(k)}(z^{(k)})$ are general convex constraints and $\mathcal{C}^{(k)}$ defines the mapping of virtual paths over physical links. This means that there could be many flows on a single virtual network, *i.e.*, a virtual network k may host (allocate) multiple services. In particular, $c_{lj}^{(k)} = 1$ if virtual path j in virtual network k uses the physical link l and 0 otherwise.⁴

The dynamism of this approach lies in the periodic bandwidth reassignment among the n_0 hosted virtual networks. The physical network in fact runs another (convex) optimization problem, whose objective is to maximize the aggregate utility of all the virtual networks, subject to some convex constraints:

$$\begin{aligned} & \text{maximize} && \sum_k w^{(k)} U^{(k)}(z^{(k)}, y^{(k)}) \\ & \text{subject to} && \mathcal{C}^{(k)} z^{(k)} \leq y^{(k)} \quad \forall k \\ & && \sum_k y^{(k)} \leq \mathcal{D} \\ & && g^{(k)}(z^{(k)}) \leq 0 \quad \forall k \\ & && z^{(k)} \geq 0 \quad \forall k \\ & \text{variables} && z^{(k)}, y^{(k)} \quad \forall k \end{aligned} \quad (4)$$

where $w^{(k)}$ is a weight (or priority) that a centralized authority in charge of embedding the slices assigns to each virtual network, and \mathcal{D} represents the physical capacities. Note how there are two levels of resource allocation in this model: each slice maximizes its utility by assigning capacity to each service hosted, and the physical network maximizes its utility by assigning resources to some slices.

As in [79], the DaVinci architecture allows (virtual) path splitting, causing packet reordering problems, and assumes the node mapping to be given. A more serious limitation is the assumption that physical links are aware of the performance objectives of all the virtual networks, which may not be possible in real world settings.

F. Distributed Virtual Network Mapping Solutions

All the previously discussed solutions assumed a centralized entity that would coordinate the mapping assignment. In other words, their solutions are limited to the intra-domain virtual network mapping. These solutions are well suited for

enterprises serving slices to their customers by using only their private resources. However, when a service must be provisioned using resources across multiple provider domains, the assumption of a complete knowledge of the substrate network becomes invalid, and another set of interesting research challenges arises.

It is well known that providers are not happy to share traffic matrices or topology information, useful for accomplishing an efficient distributed virtual network mapping. As a result, existing embedding algorithms that assume complete knowledge of the substrate network are not applicable in this scenario.

To the best of our knowledge, the first distributed virtual network mapping problem was devised by Houidi *et al.* [40]. The protocol assumes that all the requests are hub-spoke topologies, and runs concurrently three distributed algorithms at each substrate node: a *capacity-node-sorting* algorithm, a *shortest path tree* algorithm, and a *main mapping* algorithm. The first two are periodically executed to provide up to date information on node and link capacities to the main mapping.

For every element mapped, there has to be a trigger and a synchronization phase across all the nodes. The algorithm is composed of two phases: when all nodes are mapped, a shortest path algorithm is run to map the virtual links. The authors propose the use of an external signalling/control network to alleviate the problem of the heavy overhead.

In [17], the authors proposed a simultaneous node and link distributed class of mapping algorithms. In order to coordinate the node and the link mapping phases, the distributed mapping algorithm is run on the physical topology augmented with some additional logical elements (meta node and meta links) associated with the location of the physical resource.

In [16], the same authors describe a similar distributed (policy-based) inter-domain mapping protocol, based on geographic location of the physical network: PolyViNE. Each network provider keeps track of the location information of their own substrate nodes employing a hierarchical addressing scheme, and advertising availability and price information to its neighbors via a Location Awareness Protocol (LAP) — a hybrid gossiping - publish/subscribe protocol. Gossiping is used to disseminate information in a neighborhood of a network provider and pub/sub is employed so a provider could subscribe to other providers which are not in its neighborhood. PolyViNE also considers a reputation metric to cope with the lack of truthfulness in disseminating the information with the LAP protocol.

VI. ALLOCATION

Different strategies have been proposed when allocating physical resources to independent parties. Some solutions prefer practicality to efficiency, and adopt best effort approaches, (*see, e.g.*, PlanetLab [65]), while others let the (selfish) users decide the allocation outcome with a game [43], [42]. When instead it is the system that enforces the allocation, it can do it with [33] or without [5] providing guarantees. In the remainder of this section we focus first on the game theoretic solutions to resource allocation, and then on the latter case, describing first a set of solutions dealing with market-based mechanisms [5],

⁴As in [42], a system may in fact be hosted on a physical infrastructure by leasing a slice, and then provide other services by hosting (even recursively) other slices.

[49], [9], and then a reservation-based approach [33]. All those solutions focus solely on the standalone allocation task of the slice embedding problem.

A. Game-theory based allocation

Londoño *et al.* [43] defined a general pure-strategies collocation game which allows users to decide on the allocation of their requests. In their setting, customer interactions is driven by the rational behavior of users, who are free to relocate and choose whatever is best for their own interests. Under their model, a slice consists of a single node in a graph that needs to be assigned to a single resource. They define a cost function $\mathcal{K}_j(i)$ for user i when mapped to resource j as

$$\mathcal{K}_j(i) = P_j \frac{\omega_{ij}}{U_j} \quad (5)$$

where ω_{ij} is the weight (or utilization) imposed on resource j by user i , P_j is the price (in dollars) of the resource j , U_j is the overall utilization of resource j , which must satisfy its capacity constraint

$$U_j = \sum_{i \in J} \omega_i \leq \mathcal{R}_j \quad (6)$$

where J is the set of users mapped on resource j , and \mathcal{R}_j is the physical CPU capacity of resource j .

They define a rational “move” of user i from resource a to resource b if $\mathcal{R}_b(i) < \mathcal{R}_a(i)$. The game terminates when no user has a move that minimizes her cost. Note how the utility of a user (player) is higher if she can move to a more “loaded” resource, as she will share the cost with the other players hosted on the same resource.

The model has two interesting properties. First, the interaction among customers competing for resources leads to a Nash Equilibrium (NE), *i.e.* a state where no customer in the system has incentive to relocate. Second, it has been shown that the Price of Anarchy—the ratio between the overall cost of all customers under the worst-case NE and that cost under a socially optimal solution— is bounded by 3/2 and by 2 for homogeneous and heterogeneous resources, respectively. The authors also provide a generalized version of this game (General Collocation Game), in which resources to be allocated are graphs representing the set of virtual resources and underlying relationships that are necessary to support a specific user application or task. In this general case however, the equilibrium results no longer hold as the existence of a NE is not always guaranteed.

The work by Chen and Roughgarden [14] also introduces a game theoretical approach to link allocation in the form of source-destination flows on a shared network. Each flow has a weight and the cost of the link is split in proportion to the ratio between the weight of a flow and the total weights of all the flows sharing the physical link.

As shown, even recently by Chowdhury [17], in a centralized solution, the virtual network mapping problem can be thought of as a flow allocation problem where the virtual network is a flow to be allocated on a physical network.

These two game theoretic approaches may serve as inspiring example for new allocation strategies involving different

selfish principles for virtual service provisioning / competition. A system may in fact let the users play a game in which the set of strategies represent the set of different virtual networks to collocate with, in order to share the infrastructure provider costs.

B. Market-based allocation

When demand exceeds supply and not all needs can be met, virtualization systems’ goals can no longer be related to maximizing utilization, but different policies to guide resource allocation decisions have to be designed. A natural policy is to seek efficiency, namely, to allocate resources to the set of users that bring to the system the highest utility. To such an extent, the research community has frequently proposed market-based mechanisms to allocate resources among competing interests while maximizing the overall utility of the users. A subclass of solutions dealing with this type of allocation is represented by auction-based systems. An auction is the process of buying and selling goods or services by offering them up for bid, taking bids, and then selling them to the highest bidder.

Few examples where auctions have been adopted in virtualization-oriented systems are Bellagio [5], Tycoon [49] and Mirage [9]. They use a combinatorial auction mechanism with the goal of maximizing a social utility (the sum of the utilities for the users who get the resources allocated).

A *Combinatorial Auction Problem (CAP)* is equivalent to a *Set Packing Problem (SPP)*, a well studied integer program: given a set O of elements and a collection Q of subsets of these elements, with non-negative weights, SPP is the problem of finding the largest weight collection of subsets that are pairwise disjoint. This problem can be formulated as an integer program as follows: we let $\mathbf{y}_j = 1$ if the j^{th} set in W with weight w_j is selected and $\mathbf{y}_j = 0$, otherwise. Then we let $a_{ij} = 1$ if the j^{th} set in W contains element $i \in O$ and zero otherwise. If we assume also that there are b_i copies of the same element i , then we have:

$$\begin{aligned} & \text{maximize} && \sum_{j \in W} w_j \mathbf{y}_j \\ & \text{subject to} && \sum_{j \in W} a_{ij} \mathbf{y}_j \leq b_i \quad \forall i \in O \\ & && \mathbf{y}_j = \{0, 1\} \quad \forall j \in Q \end{aligned} \quad (7)$$

SPP is equivalent to a CAP if we think of the \mathbf{y}_j s as the users to be possibly allocated and requesting a subset of resources in O , and w_j as the values of their bids. Note that solving a set packing problem is NP-Hard [25]. This means that optimal algorithms to determine the winner in an auction are also NP-Hard. To deal with this complexity, many heuristics have been proposed. In [5] for example, the authors rely on a thresholding auction mechanism called SHARE [20], which uses a first-fit packing heuristic.

Another example of a system that handles the allocation for multiple users with an auction is Tycoon [49]. In Tycoon, users place bids on the different resources they need. The fraction of resource allocated to one user is her proportional share of the total bids in the system. For this reason, Tycoon’s allocation mechanism can also be considered best-effort: there are no guarantees that users will receive the desired fraction of the resources. The bidding process is continuous in the sense that

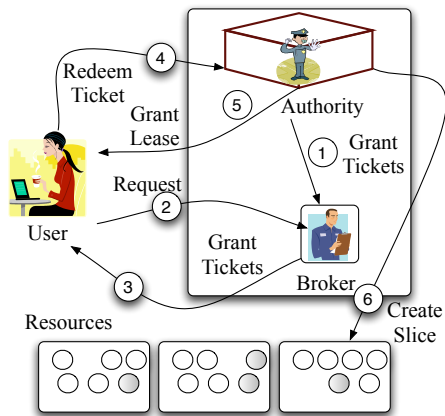


Fig. 4. Architecture and allocation phases in SHARP [33].

any user may modify or withdraw their bid at any point in time, and the allocation for all the users can be adjusted according to the new bid-to-total ratio.

As pointed out in [4], although market-based allocation systems can improve user satisfaction on large-scale federated infrastructures, and may lead to a social optimal resource allocation, there are few issues that should be taken into account when designing such mechanisms. In fact, the system may be exploited by users in many ways. Current auction-based resource allocation systems often employ very simple mechanisms, and there are known problems that may impact efficiency or fairness (see [4], Section 6). We report three of them here:

- *underbidding*: users know that the overall demand is low and they can drive the prices down.
- *iterative bidding*: often one shot auctions are not enough to reach optimal resource allocation but the iterations may not end by the time the allocations are needed.
- *auction sandwich attack*: occurs when users bid for resources in several time intervals. This attack gives the opportunity to deprive other users of resources they need, lowering the overall system utility.

C. Reservation-based allocation

As the last piece of this section on allocation approaches, we discuss a reservation-based system, SHARP [33] whose architecture is depicted in Figure 4. The system introduces a level of indirection between the user and the centralized *authority* responsible for authentication and for building the slice: the *broker or agent*. The authority issues a number of *tickets* to a number of brokers (usually many brokers responsible for a subset of resources are connected). Users then ask and eventually get tickets, and later in time, they redeem their tickets to the authority that does the final slice assignment (Figure 4).

This approach has many interesting properties but it may lead to undesirable effects. For example, coexisting brokers are allowed to split the resources: whoever has more requests should be responsible for a bigger fraction of them. This

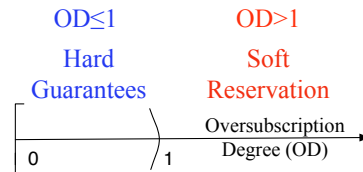


Fig. 5. Different values of Oversubscription Degree tune allocation guarantees [33].

sharing of responsibilities may bring fragmentation problems as resources become divided into many small pieces over time. Fragmentation of the resources is a weakness, as the resources become effectively unusable being divided into pieces that are too small to satisfy the current demands.

One of the most relevant contributions of SHARP in the context of the slice embedding problem, is the rule of the *Oversubscription Degree (OD)*. The *OD* is defined as the ratio between the number of issued tickets and the number of available resources. When *OD* is greater than one, *i.e.*, there are more tickets than actual available resources, the user has a probability less than one to be allocated even though she owns a ticket. When instead *OD* is less or equal than one, users with tickets have guaranteed allocation (Figure 5).

Note how the level of guarantees changes with *OD*. In particular, when the number of tickets issued by the authority increases, the level of guarantees decreases. The authors say that the allocation policy tends to a first come first serve for *OD* that tends to infinity. In other words, if there are infinite tickets, there is no reservation at all, and simply the first requests will be allocated. The oversubscription degree is not only useful to control the level of guarantees (by issuing less tickets than available resources the damage from resource loss if an agent fails or becomes unreachable is limited), but it can be used also to improve resource utilization by means of statistical multiplexing the available resources.

VII. FACILITY LOCATION PROBLEMS

In this section we discuss a set of problems similar to slice embedding: the facility location problems. Facility location is a branch of operations research whose goal is to assign a number of facilities to a set of users, while minimizing a given cost function. An ample amount of literature exists on centralized [61], [76] or distributed [32], [50] solutions for this NP-hard problem [44].

The centralized facility location problem is defined as follows: suppose we are given n potential facility locations and a list of m users who need to be serviced from these locations. There is an initial fixed cost c_i of opening the facility at location i , while there is a cost d_{ij} of serving a user j from facility i . The goal is to select (open) a set of facility locations and to assign each user to one facility, while minimizing the cost.

In order to model this problem, we define a binary decision variable z_i for each location i , which is equal to one if the facility is selected, and 0 otherwise. In addition, we define a binary variable $x_{ij} = 1$ if user j is served by facility i , and 0

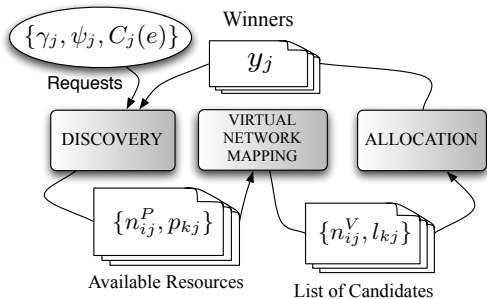


Fig. 6. Interactions and data exchanges in the slice embedding problem.

otherwise. The facility location problem is then formulated as follows:

$$\begin{aligned}
 & \text{minimize} && \sum_{i=1}^n c_i z_i + \sum_{j=1}^m \sum_{i=1}^n d_{ij} x_{ij} \\
 & \text{subject to} && \sum_{i=1}^n x_{ij} = 1 \quad \forall j \\
 & && x_{ij} \leq z_i \quad \forall i, \forall j \\
 & && x_{ij}, z_i \in \{0, 1\} \quad \forall i, \forall j.
 \end{aligned} \tag{8}$$

The affine constraint $\sum_{i=1}^n x_{ij} = 1$ enforces a single facility to a user, while the constraint $x_{ij} \leq z_i$ ensures that if there is no facility at location i , *i.e.* $z_i = 0$, then user j cannot be served there, and we must have $x_{ij} = 0$.

The facility location and the slice embedding problems may look similar since both have the high level goal of assigning a set of resources to a set of users, and both solutions require knowledge of the resource availability to work efficiently. However, the two problems differ in many aspects: first, the facility location assignment algorithms usually assume no cooperation with the discovery protocol, while in the slice embedding problem the resource discovery is directly interacting with the other two phases, as we discuss in the next section. More importantly, the slice embedding problem assumes that resources are virtual instances of both nodes and edges of the physical infrastructure, as opposed to standalone facilities to be assigned to users. This detail leads to important differences in the assignment algorithms as explained in [79] and in [52]. Moreover, facility location problems assume that each and every user has to be assigned to only one physical resource (and the positive cost to the system of such assignment is minimized), while this assumption disappears in the slice embedding problem where, in general, there may not be the guarantee that every user is allocated.

VIII. ON MODELING THE SLICE EMBEDDING PROBLEM

In this section we use optimization theory to model the interactions between the three phases of the slice embedding problem. We first model each standalone phase — resource discovery, virtual network mapping, and allocation — and subsequently model the slice embedding problem as a whole by merging the three phases into a centralized optimization problem. Consider the ellipsoid in Figure 6, augmented from Figure 1 (we explain the rest of the notation throughout this section): user j requests a virtual network composed of $\gamma_j \in \mathbb{N}$ virtual nodes, $\psi_j \in \mathbb{N}$ virtual links and a vector of constraints $C_j(e) = \langle C_j(e_1), \dots, C_j(e_c) \rangle$ where e is a vector of

$c = \gamma_j + \psi_j$ elements — nodes and links — of the network. **Discovery:** To model the resource discovery we introduce two binary variables, n_i^P and p_k that are equal to 1 if the i^{th} physical node and the k^{th} loop-free physical path, respectively, are available, and zero otherwise. An element is available if a discovery operation is able to find it, given a set of protocol parameters, *e.g.*, find all loop-free paths within a given deadline, or find as many available physical nodes as possible within a given number of hops.

If the system does not return at least γ physical nodes and ψ available loop-free physical paths among all the possible N nodes and P paths of the physical network G^P , then the user's request should be immediately discarded. Among all possible resources, the system may choose to return a set that maximizes a given notion of utility. Those utilities may have the role of selecting the resources that are closer — with respect to some notion of distance — to the given set of constraints $C(e)$. If we denote as $u_i \in \mathbb{R}$ and $\omega_k \in \mathbb{R}$ the utility of physical nodes and paths respectively, then the discovery phase of the slice embedding problem can be modeled as follows:

$$\begin{aligned}
 & \text{maximize} && f(n_i^P, p_k) = \sum_{i \in N} u_i n_i^P + \sum_{k \in P} \omega_k p_k \\
 & \text{subject to} && \sum_{i \in N} n_i^P \geq \gamma \\
 & && \sum_{k \in P} p_k \geq \psi \\
 & && n_i^P, p_k \in \{0, 1\} \quad \forall i \quad \forall k
 \end{aligned} \tag{9}$$

After the discovery phase is completed, the vectors of available physical resources (n^P, p) are passed to the virtual network mapper.

Virtual Network Mapping: This phase takes as input all the available resources (subset of all the existing resources) $P' \subseteq P$ and $N' \subseteq N$, maps virtual nodes to physical nodes, virtual links to loop-free physical paths, and returns a list of candidates — virtual nodes and virtual links — to the allocator. To model this phase, we define two sets of binary variables $n_{ij}^V \forall i \in N'$, and $l_{kj} \forall k \in P', \forall j \in J$, where J is the set of users requesting a slice. $n_{ij}^V = 1$ if a virtual instance of node i could possibly be mapped to user j and zero otherwise, while $l_{kj} = 1$ if a virtual instance of the loop-free physical path k could possibly be mapped to user j , and zero otherwise. The virtual network mapping phase of the slice embedding problem can hence be modeled by the following optimization problem:

$$\begin{aligned}
 & \text{maximize} && g(n_{ij}^V, l_{kj}) = \sum_{j \in J} (\sum_{i \in N'} \Theta_{ij} n_{ij}^V + \sum_{k \in P'} \Phi_{kj} l_{kj}) \\
 & \text{subject to} && \sum_{i \in N'} n_{ij}^V = \gamma_j \quad \forall j \in J \\
 & && \sum_{k \in P'} l_{kj} = \psi_j \quad \forall j \in J \\
 & && n_{ij}^V = n_{ij}^P \quad \forall i \in N' \quad \forall j \in J \\
 & && l_{kj} \leq p_{kj} \quad \forall k \in P' \quad \forall j \in J \\
 & && n_{ij}^P, n_{ij}^V, p_{kj}, l_{kj} \in \{0, 1\} \quad \forall i \quad \forall j \quad \forall k,
 \end{aligned} \tag{10}$$

where Θ_{ij} is the revenue that the system would get if user j gets assigned to virtual node i , and Φ_{kj} is the system's revenue if the user j gets the virtual link k . The first two constraints enforce that all the virtual resources requested by each user are mapped, the third constraint ensures that the one-to-one mapping between virtual and physical nodes is satisfied, and the fourth constraint ensures that at least one loop-free physical path is going to be assigned to each virtual link of

the requested slice.

Allocation: As soon as the virtual mapping candidates have been identified, a packing problem needs to be run, considering both user priorities and physical constraints. Enhancing the level of details from the standard set packing problem [71] to virtual nodes and links, we model the allocation phase of the slice embedding problem as follows:

$$\begin{aligned} & \text{maximize} && h(y_j) = \sum_{j \in J} w_j y_j \\ & \text{subject to} && \sum_{j \in J} n_{ij}^V y_j \leq C_i^n \quad \forall i \in N' \\ & && \sum_{j \in J} l_{kj} y_j \leq C_k^l \quad \forall k \in P' \\ & && y_j \in \{0, 1\} \quad \forall j \end{aligned} \quad (11)$$

where C_i^n and C_k^l are the number of virtual nodes and links respectively, that can be simultaneously hosted on the physical node i and physical path k , respectively, and y_j is a binary variable equal to 1 if user j has been allocated and zero otherwise. A weight w_j is assigned to each user j , and it depends on the allocation policy used (*e.g.* in first-come first-serve, $w_j = w \quad \forall j$, or in a priority based allocation w_j represents the importance of allocating user j 's request). As multiple resources are typically required for an individual slice, the slice embedding needs to invoke the appropriate resource allocation methods on individual resources, and it does so throughout this last phase. Each resource type may in fact have its own allocation policy (*e.g.*, either guaranteed or best-effort resource allocation models), and this phase only ensures that users will not be able to exceed physical limits or their authorized resource usage. For example, the system may assign a weight $w_j = 0$ to a user that has not yet been authorized, even though her virtual network could be physically mapped.

Slice Embedding: In order to clarify how the three phases of the slice embedding problem interact and how they may impact efficiency in network virtualization, we formulate a centralized optimization problem that considers the slice embedding problem as a whole. In particular, we model the three phases as follows:

$$\text{maximize} \quad \alpha \cdot f(n_{ij}^P, p_{kj}) + \beta \cdot g(n_{ij}^V, l_{kj}) + \delta \cdot h(y_j)$$

$$\text{subject to} \quad \sum_{i \in N} n_{ij}^P \geq \gamma_j \quad \forall j \quad (12a)$$

$$\sum_{k \in P} p_{kj} \geq \psi_j \quad \forall j \quad (12b)$$

$$\sum_i n_{ij}^V = \gamma_j \quad \forall j \quad (12c)$$

$$\sum_k l_{kj} = \psi_j \quad \forall j \quad (12d)$$

$$n_{ij}^V = n_{ij}^P \quad \forall i \quad \forall j \quad (12e)$$

$$l_{kj} \leq p_{kj} \quad \forall k \quad \forall j \quad (12f)$$

$$\sum_{j \in J} n_{ij}^V y_j \leq C_i^n \quad \forall i \quad (12g)$$

$$\sum_{j \in J} l_{kj} y_j \leq C_k^l \quad \forall k \quad (12h)$$

$$y_j \leq n_{ij}^V \quad \forall i \quad \forall j \quad (12i)$$

$$y_j \leq l_{kj} \quad \forall k \quad \forall j \quad (12j)$$

$$y_j, n_{ij}^P, p_{kj}, n_{ij}^V, l_{kj} \in \{0, 1\} \quad \forall i \quad \forall j \quad (12k)$$

where the first nine constraints (from (12a) to (12h)) are the same as in problems (9), (10) and (11), respectively, the two coupling constraints (12i) and (12j) guarantee that a user

is not allocated unless all the resources she queried can be mapped, and α , β and δ are normalization factors.

Note how constraints (12e), (12f) and constraints (12i) and (12j) bind the three phases of the slice embedding problem together. However, all the above constraints have never been simultaneously considered before in related literature. In [79] for example, the first two as well as the last two constraints are omitted (plus $\alpha = \delta = 0$), and a global knowledge of the resource availability is assumed. Other solutions that focus only on the virtual network mapping phase (for example [81]), omit even the capacity constraints (12g) and (12h).

From an optimization theory point of view, constraint omissions in general may result in sub-optimal solutions while constraint additions may lead to infeasible solutions. For example, the resource discovery constraints impact the other phases of the slice embedding, since a physical resource not found certainly cannot be mapped or allocated. Moreover, it is useless to run the virtual network mapping phase on resources that can never be allocated because they will exceed the physical capacity constraints. As a consequence, centralized or distributed solutions for the slice embedding problem as a whole seem to be a valuable research subarea of network virtualization.

IX. OPEN PROBLEMS

In this section we present some research challenges that are important to achieving efficient slice embedding. In general, due to its complexity, an efficient and largely scalable solution for the slice embedding problem that involves all the three tasks is still elusive.

A. Devising new heuristics and approximation algorithms

As described in Section V, the virtual network mapping is often split into node and link mappings to reduce the complexity. Note, however, that such assignments are not independent. In other words, solving them sequentially introduces sub-optimality. Researchers should therefore keep in mind that node assignments affect link assignments and vice-versa when devising heuristics for this particular task of the slice embedding problem.

Another interesting research direction is to devise heuristics for conflicting objectives. For example, it is not clear whether load balancing is the only way to improve system performance as done in [81]. One can think about optimizing other objectives such as bin packing on the physical resources to save power. Clearly these two optimization approaches are different and over the lifetime of a slice, one may need to optimize one more than the other. The *load profiling* technique presented in [59], seems to be a more generalized approach than bin packing and load balancing, where neither extreme is the objective, and the system attempts to match some target load distribution across the physical resources.

Although approximation algorithms have been discussed for similar problems (see for example [46] or in [12]), to the best of our knowledge, only in [16] they have been applied to the virtual network mapping task, thus leaving the modeling of the interaction with discovery and allocation open for further research.

B. Addressing scalability and cooperation among the slice embedding tasks

In all the solutions discussed, it is assumed that allocators have ubiquitous and updated information on the physical network. A resource allocator’s ability to make effective and efficient use of the available resources, however, is governed by how much information is available to it at the time it needs to make a decision. Thus, its interaction with the resource discovery is key. An important factor in this interaction is how much data must be passed back and forth between the two components. While passing node information—how much resources are still available on each particular physical node—should be manageable, path information is $O(n^2)$ in the number of nodes, and hence will scale poorly.

Another open question is whether and how a system can achieve efficient allocation with partial information: although we are not the first to advocate that resource discovery and allocation in virtualization oriented architectures should work tightly together (Ricci *et al.* in [68] for example, claim that the Emulab testbed is being improved by keeping this design principle in mind), it is still not clear how much data should pass between the discoverer and the allocator, how often the two tasks need to communicate, and which subset of available resources should be advertised to the allocator.

C. Modeling interactions between the slice embedding tasks

Generally, when designing solutions that involve different tasks of the slice embedding problem, researchers may utilize (distributed) optimization techniques. It is in fact possible to view each phase of the slice embedding problem as a standalone optimization problem, where different principles try to optimize the different tasks of the slice embedding problem, passing around a limited amount of information, to obtain a globally optimal embedding solution. An efficiency-overhead trade-off analysis of the mechanisms that involve such message passing among the tasks encompassing the slice embedding problem could be helpful in designing novel virtualization-based systems. Such an analysis could also be generalized to the cooperation among any coexisting infrastructure services [30], with the help of (centralized or distributed) optimization theory [8], [24], control or even game theory, for those cases where the principles involved are selfish or do not have incentives to cooperate.

D. Dissecting distributed decomposition alternatives

A systematic understanding of the decomposability structures of the slice embedding problem may help obtain the most appropriate distributed algorithms, given the application. Decomposition theory provides tools to build analytic foundations for the design of modularized and distributed control of both physical and virtual networks.

For a given problem representation, there are often many choices of distributed algorithms, each leading to different outcome of the global optimality versus message passing tradeoff [56], [64]. Which alternative is the best depends on the specifics of the slice embedding application.

We believe that qualitative or quantitative comparisons across architectural decomposition alternatives of the slice embedding problem is an interesting research area. When designing novel (virtual) network architectures for specific applications, to understand where to place functionalities and how to interface them is an issue that could be more critical than the design of how to execute and implement the functionalities themselves.

E. Supporting multiple allocators

Since each allocator can only make scheduling decisions based on the jobs submitted to it, it seems challenging to make multiple allocators work together, and this opens an interesting research direction. Allocation solutions consider only the scheduling problem, but another interesting problem is what to do *after* the resources are allocated. Since an infrastructure should be able to host customized virtual networks, each with different goals and constraints, we believe that there is not a “right” type of resource allocator, but resource allocators of modern distributed service architectures should rather support different policies for different applications that they support; for example, some users should be able to be allocated in a first come first serve manner, others should have soft or hard reservation guarantees. An architecture that would support a range of allocation policies is still missing.

F. Protocol Design and Implementation

The recently proposed distributed service architectures (*e.g.* NetServ [73] or RINA [23]) are a promising petri dish for testing novel protocols and distributed applications. In the case of RINA for example, (recursive) slice embedding protocols could be designed and prototyped over virtualization-based platforms. In particular, (inspired by [37]), we believe that designing and implementing efficient protocols to guarantee a given Service Level Agreement among slices managed by the same, or by different providers, is an interesting research area. In the case of the RINA architecture [23], where “Distributed Inter-process communication Facilities (DIF)” — the building blocks of the architecture — can be thought of as slices, this would mean designing recursive protocols to enable service provisioning across multiple tier-level providers. In fact, a DIF, just as a slice, is a service building block that can be repeated and composed in layers to build wider scoped services that meet user requirements.

Moreover, as mentioned in Section VI-A, distributed protocols to capture competition and interactions among slice embedding providers could be devised, assuming cooperation among different principles providing the service, or by means of a marketplace that allows selfish behavior.

X. CONCLUSIONS

Network virtualization has been proposed as the technology that will allow growing and testing of novel Internet architectures and protocols, overcoming the weaknesses of the current Internet, as well as testing them in repeatable and reproducible network conditions. Moreover, taking cue from current trends

in industry, it can be anticipated that virtualization will be an essential part of future networks as it allows leasing and sharing the physical (network) infrastructure. In this regard, an important challenge is the allocation of substrate resources to instantiate multiple virtual networks. In order to do so, three main steps can be identified in the so called *slice embedding problem*: resource discovery, virtual network mapping and allocation.

We outlined how these three tasks are tightly coupled, and how there exists a wide spectrum of solutions that either solve a particular task, or jointly solve multiple tasks along with the interactions between them. We then concluded with a few interesting research directions in this area.

ACKNOWLEDGMENT

We thank Azer Bestavros, John Byers, Jonathan Appavou and Karim Mattar for their valuable feedback. This work was supported in part by the National Science Foundation under grants CNS-0963974, CCF-0820138, and CNS-0720604.

REFERENCES

- [1] Jeannie Albrecht, David Oppenheimer, Amin Vahdat, and David A. Patterson. Design and Implementation Trade-offs for Wide-Area Resource Discovery. *ACM Transaction Internet Technologies*, 8(4):1–44, 2008.
- [2] David G. Andersen. Theoretical Approaches to Node Assignment. Unpublished Manuscript, December 2002.
- [3] Thomas Anderson, Larry Peterson, Scott Shenker, and Jonathan Turner. Overcoming the Internet Impasse through Virtualization. *Computer Communication ACM*, 38(4):34–41, 2005.
- [4] Alvin AuYoung, Phil Buonadonna, Brent N. Chun, Chaki Ng, David C. Parkes, Jeff Shneidman, Alex C. Snoeren, and Amin Vahdat. Two Auction-Based Resource Allocation Environments: Design and Experience. *Market Oriented Grid and Utility Computing, Rajmukar Buyya and Kris Bubendorfer (eds.), Chapter 23, Wiley, 2009.*, 2009.
- [5] Alvin AuYoung, Brent N. Chun, Alex C. Snoeren, and Amin Vahdat. Resource Allocation in Federated Distributed Computing Infrastructures. In *Proceedings of the 1st Workshop on Operating System and Architectural Support for the On-demand IT Infrastructure*, October 2004.
- [6] Suman Banerjee, Seungjoon Lee, Bobby Bhattacharjee, and Aravind Srinivasan. Resilient Multicast Using Overlays. *SIGMETRICS Perform. Eval. Rev.*, 31(1):102–113, 2003.
- [7] Andy Bavier, Nick Feamster, Mark Huang, Larry Peterson, and Jennifer Rexford. In VINI Veritas: Realistic and Controlled Network Experimentation. *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 3–14, 2006.
- [8] S. Boyd and L. Vandenberghe. *Convex Optimization*. <http://www.stanford.edu/people/boyd/cvxbook.html>, 2004.
- [9] Alvin AuYoung Chaki Ng David C. Parkes Jeffrey Shneidman Alex C. Snoeren Brent N. Chun, Philip Buonadonna and Amin Vahdat. Mirage: A Microeconomic Resource Allocation System for SensorNet Testbeds. In *Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors*, 2005.
- [10] John W. Byers, Jeffrey Considine, Michael Mitzenmacher, and Stanislav Rost. Informed Content Delivery Across Adaptive Overlay Networks. In *In Proceedings of ACM SIGCOMM*, pages 47–60, 2002.
- [11] Jorge Carapinha and Javier Jimenez. Network Virtualization—a View from the Bottom. *VISA, ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*, 17 August 2009.
- [12] Amit Chakrabarti, Chandra Chekuri, Anupam Gupta, and Amit Kumar. Approximation Algorithms for the Unsplittable Flow Problem. *APPROX '02: Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 51–66, 2002.
- [13] Kyle Chard, Kris Bubendorfer, and Peter Komisarczuk. High Occupancy Resource Allocation for Grid and Cloud Systems, a Study with DRIVE. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, pages 73–84, New York, NY, USA, 2010. ACM.
- [14] H.L. Chen and T. Roughgarden. Network Design with Weighted Players. *Theory of Computing Systems*, 45(2):302–324, 2009.
- [15] Xiang Cheng, Sen Su, Zhongbao Zhang, Hanchi Wang, Fangchun Yang, Yan Luo, and Jie Wang. Virtual Network Embedding Through Topology-Aware Node Ranking. *SIGCOMM Computer Communication Review*, 41:38–47, April 2011.
- [16] Mosharaf Chowdhury, Fady Samuel, and Raouf Boutaba. PolyViNE: Policy-Based Virtual Network Embedding across Multiple Domains. In *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, VISA '10, pages 49–56, New York, NY, USA, 2010. ACM.
- [17] N. M. Mosharaf Kabir Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Virtual Network Embedding with Coordinated Node and Link Mapping. In *INFOCOM*, pages 783–791, 2009.
- [18] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. A Survey of Network Virtualization. *Computer Networks*, 54:862–876, April 2010.
- [19] B. Chun and A. Vahdat. Workload and Failure Characterization on a Large-Scale Federated Testbed. Technical report, IRB-TR-03-040, Intel Research Berkeley., 2003.
- [20] Brent N. Chun, Chaki Ng, Jeannie Albrecht, David C. Parkes, and Amin Vahdat. Computational Resource Exchanges for Distributed Resource Allocation. 2004.
- [21] Jeffrey Considine, John W. Byers, and Ketan Meyer-Patel. A Constraint Satisfaction Approach to Testbed Embedding Services. *SIGCOMM Computer Communication Review*, 34(1):137–142, 2004.
- [22] Costas Courcoubetis and Richard R. Weber. Economic Issues in Shared Infrastructures. *VISA '09: Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pages 89–96, 2009.
- [23] John Day, Ibrahim Matta, and Karim Mattar. Networking is IPC: A Guiding Principle to a Better Internet. In *Proceedings of the 2008 ACM CoNEXT Conference*, CoNEXT '08, pages 67:1–67:6, New York, NY, USA, 2008. ACM.
- [24] D.Bertsimas and J.N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1997.
- [25] Sven de Vries and Rakesh V. Vohra. Combinatorial Auctions: A survey. *INFORMS Journal on Computing*, (3):284–309, 2003.
- [26] N. G. Duffield, Pawan Goyal, Albert Greenberg, Partho Mishra, K. K. Ramakrishnan, and Jacobus E. van der Merwe. Resource management with Hoses: Point-to-Cloud Services for Virtual Private Networks. *IEEE/ACM Transactions of Networking*, 10(5):679–692, 2002.
- [27] David Eppstein. Finding the k Shortest Paths. *SIAM J. Comput.*, 28(2):652–673, 1999.
- [28] Flavio Esposito and Ibrahim Matta. PreDA: Predicate routing for DTN architectures over MANET. In *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, pages 1–6. IEEE, November 2009.
- [29] Jinliang Fan and Mostafa H. Ammar. Dynamic Topology Configuration in Service Overlay Networks: A Study of Reconfiguration Policies. In *Proceedings of IEEE INFOCOM*, 2006.
- [30] Nick Feamster, Lixin Gao, and Jennifer Rexford. How to Lease the Internet in Your Spare Time. *SIGCOMM Computer Communication Review*, 37(1):61–64, 2007.
- [31] Global Environment for Network Innovations. <http://www.geni.net>.
- [32] Christian Frank and Kay Römer. Distributed Facility Location Algorithms for Flexible Configuration of Wireless Sensor Networks. In *Proceedings of the 3rd IEEE International Conference on Distributed Computing in Sensor Systems*, DCOSS'07, pages 124–141, Berlin, Heidelberg, 2007. Springer-Verlag.
- [33] Yun Fu, Jeffrey Chase, Brent Chun, Stephen Schwab, and Amin Vahdat. SHARP: an Architecture for Secure Resource Peering. *SIGOPS Operating System Review*, 37(5):133–148, 2003.
- [34] GENI. End-user opt-in working group <http://groups.geni.net/geni/wiki/GeniOptIn>., 2009.
- [35] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. VL2: a Scalable and Flexible Data Center Network. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, SIGCOMM '09, pages 51–62, New York, NY, USA, 2009. ACM.
- [36] Brian Hayes. Cloud computing. *Commun. ACM*, 51(7):9–11, 2008.
- [37] Jiayue He, Rui Zhang-shen, Ying Li, Cheng yen Lee, Jennifer Rexford, and Mung Chiang. DaVinci: Dynamically Adaptive Virtual Networks for a Customized Internet. In *Proc. CoNEXT*, 2008.
- [38] Mike Hibler, Robert Ricci, Leigh Stoller, Jonathon Duerig, Shashi Guruprasad, Tim Stack, Kirk Webb, and Jay Lepreau. Large-Scale Virtualization in the Emulab Network Testbed. *ATC'08: USENIX 2008 Annual Technical Conference on Annual Technical Conference*, pages 113–128, 2008.

- [39] Urs Hoelzle and Luiz Andre Barroso. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 2009.
- [40] I. Houidi, W. Louati, and D. Zeghlache. A Distributed Virtual Network Mapping Algorithm. In *Communications, 2008. ICC '08. IEEE International Conference on*, pages 5634–5640, May 2008.
- [41] An-Cheng Huang and Peter Steenkiste. Network-Sensitive Service Discovery. *USITS: USENIX Symposium on Internet Technologies and Systems*, 2003.
- [42] Vatche Ishakian, Raymond Sweha, Jorge Londoño, and Azer Bestavros. Colocation as a Service: Strategic and Operational Services for Cloud Colocation. In *NCA*, pages 76–83, 2010.
- [43] Azer Bestavros Jorge Londoño and Shanghua Teng. Collocation Games And Their Application to Distributed Resource Management. In *Proceedings of USENIX HotCloud'09: Workshop on Hot Topics in Cloud Computing, San Diego, CA., June 2009*.
- [44] O. Kariv and S. Hakimi. An Algorithmic Approach to Network Location Problems, Part II: P-Medians. *SIAM Journal on Applied Mathematics*, 37:539–560, 1979.
- [45] Morgan Kaufmann. *The Grid. Blueprint for a New Computing Infrastructure*. Elsevier Science in Grid Computing, 2 edition, December.
- [46] Stavros G. Kolliopoulos and Clifford Stein. Improved Approximation Algorithms for Unsplittable Flow Problems. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 426–435, 1997.
- [47] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach*. Addison Wesley, 2009.
- [48] C. Sansone L. P. Cordella, P. Foggia and M. Vento. An Improved Algorithm for Matching Large Graphs. *3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pages 149–159, 2001.
- [49] Kevin Lai, Lars Rasmusson, Eytan Adar, Li Zhang, and Bernardo A. Huberman. Tycoon: An Implementation of a Distributed, Market-Based Resource Allocation System. *Multiaigent Grid Syst.*, 1(3):169–182, 2005.
- [50] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis, and A. Bestavros. Distributed Placement of Service Facilities in Large-Scale Networks. In *INFOCOM*, Anchorage, AK, May 2007.
- [51] Harold C. Lim, Shivnath Babu, Jeffrey S. Chase, and Sujay S. Parekh. Automated Control in Cloud Computing: Challenges and Opportunities. In *Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds, ACDC '09*, pages 13–18, New York, NY, USA, 2009. ACM.
- [52] Jens Lischka and Holger Karl. A Virtual Network Mapping Algorithm based on Subgraph Isomorphism Detection. *VISA, ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures*, pages 81–88, 2009.
- [53] Michael Litzkow, Miron Livny, and Matthew Mutka. Condor - A Hunter of Idle Workstations. *Proceedings of the 8th International Conference of Distributed Computing Systems*, June 1988.
- [54] Jorge Londoño and Azer Bestavros. NETEMBED: A Network Resource Mapping Service for Distributed Applications. *Symposium on Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International*, pages 1–8, April 2008.
- [55] Jing Lu and Jonathan Turner. Efficient Mapping of Virtual Networks onto a Shared Substrate. Technical report, Washington University in St. Louis, 2006.
- [56] R.A. Calderbank M. Chiang, S.H. Low and J.C. Doyle. Layering as Optimization Decomposition: A Mathematical Theory of Network Architectures. *Proc. of IEEE*, 95(1):255–312, Jan 2007.
- [57] Harsha V. Madhyastha, Ethan Katz-bassett, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. iplane nano: Path prediction for peer-to-peer applications. *Proceedings of NSDI*, 2009.
- [58] Matthew L. Massie, Brent N. Chun, and David E. Culler. The Ganglia Distributed Monitoring System: Design, Implementation and Experience. *Parallel Computing*, 30:2004, 2003.
- [59] Ibrahim Matta and Azer Bestavros. A Load Profiling Approach to Routing Guaranteed Bandwidth Flows. *INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 3:1014–1021 vol.3, mar-2 apr 1998.
- [60] Elena Meshkova, Janne Riihijärvi, Marina Petrova, and Petri Mähönen. A Survey on Resource Discovery Mechanisms, Peer-to-Peer and Service Discovery Frameworks. *Computer Networks*, 52(11):2097–2128, 2008.
- [61] P. Mirchandani and R. Francis. *Discrete Location Theory*. Wiley, 1990.
- [62] Albrecht J. Patterson D. Vahdat A. Oppenheimer, D. Design and Implementation Tradeoffs for Wide-Area Resource Discovery. *High Performance Distributed Computing, 2005. HPDC-14. Proceedings. 14th IEEE International Symposium on*, pages 113 – 124, July 2005.
- [63] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, November 1999.
- [64] D.P. Palomar and Mung Chiang. A Tutorial on Decomposition Methods for Network Utility Maximization. *Selected Areas in Communications, IEEE Journal on*, 24(8):1439–1451, aug. 2006.
- [65] Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. *SIGCOMM Computer Communication Review*, 33(1):59–64, 2003.
- [66] BGP/MPLS RFC2547. <http://tools.ietf.org/html/rfc2547>.
- [67] Robert Ricci, Chris Alfeld, and Jay Lepreau. A Solver for the Network Testbed Mapping Problem. *SIGCOMM Computer Communication Review*, 33(2):65–81, 2003.
- [68] Robert Ricci, David Oppenheimer, Jay Lepreau, and Amin Vahdat. Lessons from Resource Allocators for Large-Scale Multiuser Testbeds. *ACM SIGOPS Operating Systems Review*, 40(1), January 2006.
- [69] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, Get Off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds. In *Proceedings of the 16th ACM conference on Computer and communications security, CCS '09*, pages 199–212, New York, NY, USA, 2009. ACM.
- [70] Jeffrey Shneidman, Chaki Ng, David C. Parkes, Alvin AuYoung, Alex C. Snoeren, Amin Vahdat, and Brent Chun. Why Markets Would (But Don't Currently) Solve Resource Allocation Problems in Systems. In *Proceedings of the 10th conference on Hot Topics in Operating Systems - Volume 10*, pages 7–7, Berkeley, CA, USA, 2005. USENIX Association.
- [71] Steven S. Skiena. *Set Packing. The Algorithm Design Manual*. 1997.
- [72] David Spence and Tim Harris. XenoSearch: Distributed Resource Discovery in the XenoServer Open Platform. *International Symposium on High-Performance Distributed Computing (HPDC)*, page 216, 2003.
- [73] Eric Liu Mike Kester Henning Schulzrinne Volker Hilt Srin Seetharaman Suman Srinivasan, Jae Woo Lee and Ashiq Khan. NetServ: Dynamically Deploying In-network Services. In *Proceedings of ReArch '09 (CoNEXT workshop)*, 2009.
- [74] SWORD. Source code <http://sword.cs.williams.edu/>, 2005.
- [75] Jonathan Turner and David Taylor. Diversifying the Internet. *GLOBE-COM IEEE Global Communication conference*, 2005.
- [76] J Vygen. Approximation Algorithms for Facility Location Problems. Technical report, 05950-OR, Res. Ins. for Disc. Math., University of Bonn, 2005.
- [77] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):255–270, 2002.
- [78] Jon Whiteaker, Fabian Schneider, and Renata Teixeira. Explaining packet delays under virtualization. *ACM SIGCOMM CCR*, 41(1):38–44, January 2011.
- [79] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration. *SIGCOMM Computer Communication Review*, 38(2):17–29, 2008.
- [80] Tao Yu and Kwei-Jay Lin. A Broker-Based Framework for QoS-Aware Web Service Composition. In *IEEE '05: Proceedings of the 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'05) on e-Technology, e-Commerce and e-Service*, pages 22–29, Washington, DC, USA, 2005. IEEE Computer Society.
- [81] Ammar M. Zhu, Y. Algorithms for Assigning Substrate Network Resources to Virtual Network Components. *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, pages 1–12, April 2006.
- [82] Qian Zhu and Gagan Agrawal. Resource Provisioning with Budget Constraints for Adaptive Applications in Cloud Environments. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC '10*, pages 304–307, New York, NY, USA, 2010. ACM.