Competitive and Deterministic Embeddings of Virtual Networks

Guy Even[1], Moti Medina[1], Gregor Schaffrath[2], Stefan Schmid[2]
[1] Tel Aviv University, {guy,medinamo}@eng.tau.ac.il
[2] Telekom Innovation Laboratories (T-Labs) & TU Berlin
{grsch,stefan}@net.t-labs.tu-berlin.de

Competitive and Deterministic Embeddings of Virtual Networks

Guy Even[1], Moti Medina[1], Gregor Schaffrath[2], Stefan Schmid[2]
[1] Tel Aviv University, {guy,medinamo}@eng.tau.ac.il
[2] Telekom Innovation Laboratories (T-Labs) & TU Berlin
{grsch,stefan}@net.t-labs.tu-berlin.de

Abstract

Network virtualization is an important concept to overcome the ossification of today's Internet as it facilitates innovation also in the network core and as it promises a more efficient use of the given resources and infrastructure. Virtual networks (VNets) provide an abstraction of the physical network: multiple VNets may cohabit the same physical network, but can be based on completely different protocol stacks (also beyond IP). One of the main challenges in network virtualization is the efficient admission control and embedding of VNets. The demand for virtual networks (e.g., for a video conference) can be hard to predict, and once the request is accepted, the specification / QoS guarantees must be ensured throughout the VNet's lifetime. This requires an admission control algorithm which only selects high-benefit VNets in times of scarce resources, and an embedding algorithm which realizes the VNet in such a way that the likelihood that future requests can be embedded as well is maximized.

This article describes a generic algorithm for the online VNet embedding problem which does not rely on any knowledge of the future VNet requests but whose performance is competitive to an optimal offline algorithm that has complete knowledge of the request sequence in advance: the so-called competitive ratio is, loosely speaking, logarithmic in the sum of the resources. Our algorithm is generic in the sense that it supports multiple traffic models, multiple routing models, and even allows for nonuniform benefits and durations of VNet requests.

Keywords: Network virtualization, online algorithms, primal-dual approach, linear programming

1. Introduction

Virtualization is an attractive design principle as it abstracts heterogeneous resources and as it allows for resource sharing. Over the last years, *end-system virtualization* (e.g., Xen or VMware) revamped the server business, and we witness a trend towards *link-virtualization*: router vendors such as Cisco and Juniper offer router virtualization, and Multiprotocol Label Switching (MPLS) solutions and Virtual Private Networks (VPNs) are widely deployed. Also split architectures like OpenFlow receive a lot of attention as they open new possibilities to virtualize links.

Network virtualization [15] goes one step further and envisions a world where multiple *virtual networks (VNets)*—which can be based on different networking protocols—cohabit the same physical network (the so-called *substrate network*). VNet requests are issued to a network provider and can have different specifications, in terms of Quality-of-Service (QoS) requirements, supported traffic and routing models, duration, and so on. The goal of the provider is then to decide whether to accept the request and at what price (*admission control*), and subsequently to realize (or *embed*) the VNet such that its specification is met while minimal resources are used—in order to be able to accept future requests.

Virtual networks have appealing properties, for instance, (1) they allow to innovate the Internet by making the network core "programmable" and by facilitating service-tailored networks which are optimized for the specific application (e.g., content distribution requires different technologies and QoS guarantees than, live streaming, gaming, or online social networking); (2) the given resources can be (re-)used more efficiently, which saves cost at the provider side; (3) start-up companies can experiment with new protocols and services without investing in an own and expensive infrastructure; among many more.

Due to the flexibility offered by network virtualization, the demand for virtual networks can be hard to predict—both in terms of arrival times and VNet durations. For example, a VNet may be requested at short notice for a video conference between different stakeholders of an international project. It is hence mandatory that this VNet be realized quickly (i.e., the admission and embedding algorithms must have low time complexities) and that sufficient resources are *reserved* for this conference (to ensure the QoS spec).

This article attends to the question of how to handle VNets arriving one-by-one in an *online fashion* [9]: Each request either needs to be embedded or rejected, and the online setting means that the decision (embed or reject) must be taken without any information about future requests; the decision cannot be changed later (no preemption).

The goal is to maximize the overall profit, i.e., the sum of the benefits of the embedded VNets. We use competitive analysis for measuring the quality of our online algorithm. The *competitive ratio* of an online algorithm is α if, for every sequence of requests σ, the benefit obtained by the algorithm is at least an α fraction of the optimal offline benefit, that is, the benefit obtainable by an algorithm with complete knowledge of the request sequence σ in advance.

1.1. VNet Specification and Service Models

There are many service models for VNets [26], and we seek to devise generic algorithms applicable to a wide range of models. The two main aspects of a service model concern the modeling of traffic and the modeling of routing.

Traffic. We briefly outline and compare three models for allowable traffic. (1) In the *customer-pipe model*, a request for a VNet includes a traffic matrix that specifies the required bandwidth between every pair of terminals of the VNet. (2) In the *hose model* [16, 21], each terminal v is assigned a maximum ingress bandwidth $b_{in}(v) \geq 1$ and a maximum egress bandwidth $b_{out}(v) \geq 1$. Any traffic matrix that is consistent with the ingress/egress values must be served, (3) Finally, we propose an *aggregate ingress model*, in which the set of allowed traffic patterns is specified by a single parameter $\mathcal{I} \geq 1$. Any traffic in which the sum of ingress bandwidths is at most $\mathcal{I}$ must be served.

The customer-pipe model sets detailed constraints on the VNet and enables efficient utilization of network resources as the substrate network has to support only a single traffic matrix per VNet. On the other hand, the hose model offers flexibility since the allowed traffic matrices constitute a polytope. Therefore, the VNet embedding must to take into account the "worst" allowable traffic patterns.

Multicast sessions are not efficiently supported in the customer-pipe model and the hose model. In these models, a multicast session is translated into a set of unicasts from the ingress node to each of the egress nodes. Thus, the ingress bandwidth of a multicast is multiplied by the number of egress nodes [17, 18, 22, 24].

In the aggregate ingress model, the set of allowable traffic patterns is wider, offers simpler specification, and more flexibility compared to the hose model. In addition, multicasting and broadcasting do not incur any penalty at all since intermediate nodes in the substrate network duplicate packets exiting via different links instead of having multiple duplicates input by the ingress node. For example, the following traffic patterns are allowed in the aggregate ingress model with parameter $\mathcal{I}$: (i) a single multicast from one node with bandwidth $\mathcal{I}$, and (ii) a set of multicast sessions with bandwidths f_i, where $\sum_i f_i \leq \mathcal{I}$. Hence, in the aggregate ingress model traffic may vary from a "heavy" multicast (e.g., software update to multiple branches) to a multi-party video-conference session in which every participant multicasts her video and receives all the videos from the other participants.

Routing. We briefly outline three models for the allowed routing. (1) In *tree routing*, the VNet is embedded as a Steiner tree in the substrate network that spans the terminals of the VNet. (2) In *single path routing*, the VNet is embedded as a union of paths between every pair of terminals. Each pair of terminals communicates along a single path. (3) In *multipath routing*, the VNet is embedded as a union of linear combinations of paths between terminals. Each pair of terminals u and v communicates along multiple paths. The traffic from node u to node v is split among these paths. The linear combination specifies how to split the traffic.

In tree routing and single path routing, all the traffic between two terminals of the same VNet traverses the same single path. This simplifies routing and keeps the packets in order. In multipath routing, traffic between two terminals may be split between multiple paths. This complicates routing since a router needs to decide through which port a packet should be sent. In addition, routing tables are longer, and packets may arrive out of order. Finally, multicasting with multipath routing requires network coding [1].

Packet Rate. We consider link bandwidth as the main resource of a link. However, throughput can also depend on processing power of the network nodes. Since a router needs to inspect each packet to determine its actions, the load incurred on a router is mainly influenced by the so-called *packet rate*, which we model as an additional parameter of a VNet request. If packets have uniform length, then the packet rate is a linear function of the bandwidth.

Duration and Benefit. The algorithms presented in this article can be competitive with respect to the *total number* of embedded VNets. However, our approach also supports a more general model where VNets have *different* benefits. Moreover, we can deal with VNets of finite durations. Therefore, in addition to the specification of the allowable traffic patterns, each request for a VNet has the following parameters: (i) *duration*, i.e., the start and finish times of the request, and (ii) *benefit*, i.e., the revenue obtained if the request is served.

1.2. Previous Work

For an introduction and overview of network virtualization, the reader is referred to [15]. A description of a prototype network virtualization architecture (under development at Telekom Innovation Laboratories) appears in [31].

The virtual network embedding problem has already been studied in various settings, and it is well-known that many variants of the problem are computationally hard (see, e.g., [2, 14]). Optimal embeddings in the multi-path routing model exist for all traffic models. In fact, in the customer pipe model, an optimal multipath fractional embedding can be obtained by solving a multicommodity flow problem. In the hose model, an optimal reservation for multipath routing in the hose model is presented in [18]. This algorithm can also be modified to handle the aggregate ingress model.

Offline algorithms for tree routing and single path routing lack edge capacities and have only edge flow costs. Namely, these algorithm approximate a min-cost embedding in the substrate graph without capacity constraints. In the hose model, constant approximation algorithms have been developed for tree routing [17, 22, 24]. In the special case that the sum of the ingresses equals the sum of the egresses, an optimal tree can be found efficiently, and the cost of an optimal tree is within a factor three of the cost of an optimal reservation for multipath routing [25, 29]. Kumar et. al [29] proved that in presence of edge capacity constraints, computing a tree routing in the hose model is NP-hard even in the case where $b_{in}(v) = b_{out}(v)$, for every node v. Moreover, they also showed approximating the optimal tree routing within a constant factor is NP-hard.

Published online algorithms for VNet embeddings are scarce. In [23, 30], an online algorithm for the hose model with tree routing is presented. The algorithm uses a pruned BFS tree as an oracle. Edge costs are the ratio between the demand and the residual capacity. We remark that, even in the special case of online virtual circuits ("call admission"), using such linear edge costs lead to trivial linear competitive ratios [5]. The rejection ratio of the algorithm is analyzed in [23, 30], but not the competitive ratio. The problem of embedding multicast requests in an online setting was studied in [28]. They used a heuristic oracle that computes a directed Steiner tree. The competitive ratio of the algorithm in [28] is not studied. In fact, much research has focused on heuristic approaches, e.g., [20] proposes heuristic methods for constructing different flavors of reconfiguration policies; and [34] proposes subdividing heuristics and adaptive optimization strategies to reduce node and link stress. In [4], an online algorithm is presented for the case of multiple multicast requests in which the terminals the requests arrive in an arbitrarily interleaved order. The competitive ratio of the online algorithm in [4] is $O(\log n \cdot \log d)$, where n denotes the number of nodes in the substrate network and d denotes the diameter of the substrate network.

Bansal et al. [8] presented a result on network mapping in cloud environments where the goal is to minimize congestion induced by the embedded workloads, i.e., to minimize the edge capacity augmentation w.r.t. a feasible optimal embedding. They consider two classes of workloads, namely depth-d trees and complete-graph workloads, and describe an online algorithm whose competitive ratio is logarithmic in the number of substrate resources, i.e., nodes and edges. Moreover, every node in the workload is mapped to a node in the substrate network, and every edge is mapped to a single path in the substrate network between its (mapped) nodes. In contrast, we allow arbitrary workloads, a wide range of traffic models and routing models, specify the mapping of nodes, and focus is on revenue maximization.

Circuit switching is a special case of VNet embeddings in which each VNet consists of two terminals and the routing is along a single path. Online algorithms for maximizing the revenue of circuit switching were presented in [5]. A general primal-dual setting for online packing and covering appears in [10, 12]. In the context of circuit switching, the *load* of an edge e in a network is the ratio between the capacity reserved for the paths that traverses e, and the capacity of e. In the case of load (or congestion) minimization the online algorithm competes with the minimum augmentation [3, 6, 10, 8]. In the case of permanent routing, Aspnes et. al [3] designed an algorithm that augments the edge capacities by a factor of at most $O(\log n)$ w.r.t. a feasible optimal routing. Aspnes et. al [3] also showed how to use approximated oracles to embed min-cost Steiner trees in the context of multicast virtual circuit routing.

1.3. Our Contribution

This article describes an algorithmic framework called GVOP (for *general VNet online packing algorithm*) for online embeddings of VNet requests. This framework allows us to decide in an online fashion whether the VNet should be admitted or not. For the embedding itself, an *oracle* is assumed which computes the embeddings of VNets. While our framework yields fast algorithms, the embedding itself may be computationally hard and hence approximate oracles maybe be preferable in practice. We provide an overview of the state-of-the-art approximation algorithms for the realization of these oracles, and we prove that the competitive ratio is not increased much when approximate oracles are used in GVOP. Our framework follows the primal-dual online packing scheme by Buchbinder and Naor [10, 12] that provides an explanation of the algorithm of Awerbuch et al. [5].

In our eyes, the main contribution of this article lies in the generality of the algorithm in terms of supported traffic and routing models. The GVOP algorithm is input VNet requests from multiple traffic models (i.e., customer-pipe, hose , or aggregate ingress models) and multiple routing models (i.e., multipath, single path, or tree routing). This implies that the network resources can be shared between requests of all types.

We prove that the competitive ratio of our deterministic online algorithm is, in essence, logarithmic in the resources of the network. The algorithm comes in two flavors: (i) A bi-criteria algorithm that achieves a constant fraction of the optimal benefit while augmenting resources by a logarithmic factor. Each request in this version is either fully served or rejected. (ii) An online algorithm that achieves a logarithmic competitive ratio without resource augmentation. However, this version may serve a fraction of a request, in which case the associated benefit is also the same fraction of the benefit of the request. However, if the allowed traffic patterns of a request consume at most a logarithmic fraction of every resource, then this version either rejects the request or fully embeds it.

1.4. Article Organization

The remainder of this article is organized as follows. We introduce the formal model and problem definition in Section 2. The main result is presented in Section 3. The algorithmic framework is described in Section 4. Section 5 shows how to apply the framework to the VNet embedding problem and discusses the embedding oracles to be used in our framework under the different models. The article concludes with a short discussion in Section 6.

2. Problem Definition

We assume an undirected communication network $G = (V, E)$ (called the *physical network* or the *substrate network*) where V represents the set of substrate nodes (or routers) and E represents the set of links. Namely, $\{u, v\} \in E$ for $u, v \in V$ denotes that u is connected to v by a communication link. Each edge e has a capacity $c(e) \geq 1$. In Section 5.2, we will extend the model also to node capacities (processing power of a node, e.g., to take into account router loads).

The online input is as follows. The operator (or provider) of the substrate network G receives a sequence of VNet requests $\sigma = \{r_1, r_2 \ldots\}$. Upon arrival of request r_j, the operator must either reject r_j or embed it. A request r_j and the set of valid embeddings of r_j depend on the service model. A VNet request r_j has the following parameters: (1) A set $U_j \subseteq V$ of terminals, i.e., the nodes of the VNet. (2) A set Tr_j of allowed traffic patterns between the terminals. For example, in the customer-pipe model, Tr_j consists of a single traffic matrix. In the hose model, Tr_j is a polytope of traffic matrices. (3) The routing model (multipath, single path, or tree). (4) The benefit b_j of r_j. This is the revenue if the request is fully served. (5) The duration $T_j = [t_j^{(0)}, t_j^{(1)}]$ of the request. Request r_j arrives and starts at time $t_j^{(0)}$ and ends at time $t_j^{(1)}$.

The set of valid embeddings of a VNet request r_j depends on the set Tr_j of allowed traffic patterns, the routing model, and the edge capacities, for example: (1) In the customer-pipe model with multipath routing, an embedding is a multicommodity flow. (2) In the hose model with tree routing, a valid embedding is a set of edges with bandwidth reservations that induces a tree that spans the terminals. The reserved bandwidth on each edge may not exceed the capacity of the edge. In addition, the traffic must be routable in the tree with the reserved bandwidth.

If the allowed traffic patterns of a request r_j consume at most a logarithmic fraction of every resource, then our algorithm either rejects the request or fully embeds it. If a request consumes at least a logarithmic fraction of the resources, then the operator can accept and embed a fraction of a request. If an operator accepts an ϵ-fraction of r_j, then this means that it serves an ϵ-fraction of every allowed traffic pattern. For example, in the customer-pipe model with a traffic matrix Tr, only the traffic matrix $\epsilon \cdot Tr$ is routed. The benefit received for embedding an ϵ-fraction of r_j is $\epsilon \cdot b_j$. The goal is to maximize the sum of the received benefits.

3. The Main Result

Consider an embedding of VNet requests. We can assign two values to the embedding: (1) The benefit, namely, the sum of the benefits of the embedded VNets. (2) The maximum congestion of a resource. The congestion of a resource is the ratio between the load of the resource and the capacity of a resource. For example, the load of an edge is the flow along the edge, and the usage of a node is the rate of the packets it must inspect. A bi-criteria competitive online packing algorithm is defined as follows.

Definition 1. *Let OPT denote an optimal offline fractional packing solution. An online packing algorithm Alg is (α, β)-competitive if: (i) For every input sequence*

σ, the benefit of $Alg(\sigma)$ is at least $1/\alpha$ times the benefit of OPT. (ii) For every input sequence σ and for every resource e, the congestion incurred by $Alg(\sigma)$ is at most β.

The main result of this article is formulated in the following theorem. Consider a sequence of VNet requests $\{r_j\}_j$ that consists of requests from one of the following types: (i) customer pipe model with multipath routing, (ii) hose model with multipath routing, or single path routing, or tree routing, or (iii) aggregate ingress model with multipath routing, or single path routing, or tree routing.

Theorem 1. *Let $\beta = O(\log(|E| \cdot (\max_e c_e) \cdot (\max_j b_j)))$. For every sequence $\{r_j\}_j$ of VNet requests, our GVOP algorithm is a $(2, \beta)$-competitive online all-or-nothing VNet embedding algorithm.*

Note that the competitive ratio does not depend on the number of requests.
The proof of Theorem 1 appears in Sections 4 and 5.

4. A Framework for Online Embeddings

Our embedding framework is an adaptation of the online primal-dual framework by Buchbinder and Naor [11, 12]. We allow VNet requests to have finite durations and introduce approximate oracles which facilitate faster but approximate embeddings. In the following, our framework is described in detail.

4.1. LP Formulation

In order to devise primal-dual online algorithms, the VNet embedding problem needs to be formulated as a *linear program (LP)*. Essentially, a linear program consists of two parts: a linear objective function (e.g., minimize the amount of resources used for the embedding), and a set of constraints (e.g., VNet placement constraints). As known from classic approximation theory, each linear program has a corresponding *dual formulation*. The primal LP is often referred to as the *covering problem*, whereas the dual is called the *packing problem*. In our online environment, we have to deal with a dynamic *sequence* of such linear programs, and our goal is to find good approximate solutions over time [11, 12].

In order to be consistent with related literature, we use the motivation and formalism from the online *circuit switching problem* [5] (with permanent requests). Let $G = (V, E)$ denote a graph with edge capacities c_e. Each request r_j for a virtual circuit is characterized by the following parameters: (i) a source node $a_j \in V$ and a destination $dest_j \in V$, (ii) a bandwidth demand d_j, (iii) a benefit b_j. Upon arrival of a request r_j, the algorithm either rejects it or fully serves it by reserving a bandwidth of d_j along a path from a_j to *dest*$_j$. We refer to such a solution as *all-or-nothing*. The algorithm may not change previous decisions. In particular, a rejected request may not be served later, and a served request may not be rerouted or stopped (even if a lucrative new request arrives). A solution must not violate edge capacities, namely, the sum of the bandwidths reserved along each edge e is at most c_e. The algorithm competes with an optimal fractional solution that may partially serve a request using multiple paths. The optimal solution is offline, i.e., it is computed based on full information of all the requests.

First, let us devise the linear programming formulation of the dual, i.e., of *online packing*. Again, to simplify reading, we use the terminology of the online circuit switching problem with durations. Let Δ_j denote the set of valid embeddings of r_j (e.g., Δ_j is the set of paths from a_j to *dest*$_j$ with flow d_j). Define a dual variable $y_{j,\ell} \in [0, 1]$ for every "satisfying flow" $f_{j,\ell} \in \Delta_j$. The variable $y_{j,\ell}$ specifies what fraction of the flow $f_{j,\ell}$ is reserved for request r_j. Note that obviously, an application of our framework does not require an explicit representation of the large sets Δ_j (see Section 5).

Online packing is a *sequence of linear programs*. Upon arrival of request r_j, the variables $y_{j,\ell}$ corresponding to the "flows" $f_{j,\ell} \in \Delta_j$ are introduced. Let Y_j denote the column vector of dual variables introduced so far (for requests $r_1, \ldots, r_j$). Let B_j denote the benefits column vector $(b_{1,1}, \ldots, b_{1,|\Delta_1|}, \ldots, b_{j,1}, \ldots, b_{j,|\Delta_j|})^T$, where $\forall \ell, k : b_{i,\ell} = b_{i,k}$ for every i, hence we abbreviate and refer to $b_{i,\ell}$ simply as b_i. Let C denote the "capacity" column vector $(c_1, \ldots, c_N)^T$, where N denotes the number of "edges" (or resources in the general case). The matrix A_j defines the "capacity" constraints and has dimensionality $N \times \sum_{i \leq j} |\Delta_i|$. An entry $(A_j)_{e,(i,\ell)}$ equals the flow along the "edge" e in the "flow" $f_{i,\ell}$. For example, in the case of circuit switching, the flow along an edge e by $f_{i,\ell}$ is d_i if e is in the flow path, and zero otherwise. In the general case, we require that every "flow" $f_{j,\ell}$ incurs a positive "flow" on at least one "edge" e. Thus, every column of A_j is nonzero. The matrix A_{j+1} is an augmentation of the matrix A_j, i.e., $|\Delta_{j+1}|$ columns are added to A_j to obtain A_{j+1}. Let D_j denote a 0-1 matrix of dimensionality $j \times \sum_{i \leq j} |\Delta_i|$. The matrix D_j is a block matrix in which $(D_j)_{i,(i',\ell)} = 1$ if $i = i'$, and zero otherwise. Thus, D_{j+1} is an augmentation of D_j; in the first j rows, zeros are added in the new $|\Delta_{j+1}|$ columns, and, in row $j + 1$, there are zeros in the first $\sum_{i \leq j} |\Delta_i|$ columns, and ones in the last $|\Delta_{j+1}|$ columns. The matrix D_j defines the "demand" constraints. The packing linear program (called the dual LP) and the corresponding primal covering LP are listed in Figure 1. The covering LP has two variable vectors X and Z_j. The vector X has a component

$$\begin{aligned} \min Z_j^T \cdot \vec{1} + X^T \cdot C \;\; s.t. \\ Z_j^T \cdot D_j + X^T \cdot A_j \geq B_j^T \\ X, Z_j \geq \vec{0} \end{aligned}$$	$$\begin{aligned} \max B_j^T \cdot Y_j \;\; s.t. \\ A_j \cdot Y_j \leq C \\ D_j \cdot Y_j \leq \vec{1} \\ Y_j \geq \vec{0} \end{aligned}$$
(I)	(II)

Figure 1: (I) The primal covering LP. (II) The dual packing LP.

x_e for each "edge" e. This vector should be interpreted as the cost vector of the resources. The variable Z_j has a component z_i for every request r_i where $i \leq j$.

4.2. Generic Algorithm

This section presents our online algorithm GVOP to solve the dynamic linear programs of Figure 1. The formal listing appears in Algorithm 1.

We assume that all the variables X, Z, Y, (i.e, primal and dual) are initialized to zero (using lazy initialization). Since the matrix A_{j+1} is an augmentation of A_j, we abbreviate and refer to A_j simply as A. Let $\text{col}_{(j,\ell)}(A)$ denote the column of A (in fact, A_j) that corresponds to the dual variable $y_{j,\ell}$. Let $\gamma(j,\ell) \triangleq X^T \cdot \text{col}_{(j,\ell)}(A)$, where the values of X^T are with respect to the end of the processing of request r_{j-1}. It is useful to interpret $\gamma(j,\ell)$ as the X-cost of the "flow" $f_{j,\ell}$ for request j. Let $w(j,\ell) \triangleq \vec{1}^T \cdot \text{col}_{j,\ell}(A)$, namely, $w(j,\ell)$ is the sum of the entries in column (j,ℓ) of A. Since every column of A is nonzero, it follows that $w(j,\ell) > 0$ (and we may divide by it).

Definition 2. *Let Y^* denote an optimal offline fractional solution. A solution $Y \geq 0$ is (α,β)-competitive if: (i) For every j, $B_j^T \cdot Y_j \geq \frac{1}{\alpha} \cdot B_j^T \cdot Y_j^*$. (ii) For every j, $A_j \cdot Y_j \leq \beta \cdot C$ and $D_j \cdot Y_j \leq \vec{1}$.*

The following theorem can be proved employing the techniques of [11].

Theorem 2. *Assume that: (i) for every row e of A, $\max_{j,\ell} A_{e,(j,\ell)} \leq c_e$, (ii) for every row e of A, $\min_{j,\ell} A_{e,(j,\ell)} \in \{0\} \cup [1,\infty)$, (iii) for every column (i,ℓ) of A, $w(i,\ell) > 0$, and (iv) $\min_j b_j \geq 1$. Let $\beta \triangleq \log_2(1+3\cdot(\max_{j,\ell} w(j,\ell))\cdot(\max_j b_j))$. The GVOP algorithm is a $(2,\beta)$-competitive online all-or-nothing VNet packing algorithm.*

Algorithm 1 The General all-or-nothing VNet Packing Online Algorithm (GVOP).

Upon arrival of request r_j:

1. $f_{j,\ell} \leftarrow$ argmin$\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$ (oracle procedure)
2. If $\gamma(j,\ell) < b_j$, then **accept** r_j:
   (a) $y_{j,\ell} \leftarrow 1$.
   (b) $z_j \leftarrow b_j - \gamma(j,\ell)$.
   (c) For each row e do:
   $$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$
3. Else **reject** r_j (note that X, Z_j, Y_j have not been changed.)

Proof. Let us denote by *Primal*$_j$ (respectively, *Dual*$_j$) the change in the primal (respectively, dual) cost function when processing request j.

We show that *Primal*$_j \leq 2 \cdot$ *Dual*$_j$ for every j. We prove that GVOP produces feasible primal solutions throughout its execution. Initially, the primal and the dual solutions are 0, and the claim holds. Let $x_e^{(j)}$ denote the value of the primal variable x_e when r_j is processed, i.e., after Step 2c and before the execution of Step 2c upon the arrival of request r_{j+1}, in particular $x_e^{(0)} = 0$ for every e. If r_j is rejected then *Primal*$_j$ $=$ *Dual*$_j$ $= 0$ and the claim holds. Then for each accepted request r_j, *Dual*$_j = b_j$ and *Primal*$_j = \sum_{e \in E(j,\ell)} (x_e^{(j)} - x_e^{(j-1)}) \cdot c_e + z_j$, where $E(j,\ell) = \{e \in \{1,\ldots,N\} : A_{e,(j,\ell)} \neq 0\}$. Step 2c increases the cost $X^T \cdot C = \sum_e x_e \cdot c_e$ as follows:

$$\begin{aligned}
\sum_{e \in E(j,\ell)} (x_e^{(j)} - x_e^{(j-1)}) \cdot c_e &= \sum_{e \in E(j,\ell)} \left[x_e^{(j-1)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1) + \frac{1}{w(j,\ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1) \right] \cdot c_e \\
&= \sum_{e \in E(j,\ell)} \left(x_e^{(j-1)} + \frac{1}{w(j,\ell)} \right) \cdot (2^{A_{e,(j,\ell)}/c_e} - 1) \cdot c_e \\
&\leq \sum_{e \in E(j,\ell)} \left(x_e^{(j-1)} + \frac{1}{w(j,\ell)} \right) \cdot A_{e,(j,\ell)} = \gamma(j,\ell) + 1 \,.
\end{aligned}$$

where the third inequality holds since $\max_{j,\ell} A_{e,(j,\ell)} \leq c_e$. Hence after Step 2b:

$$\textit{Primal}_j \;\leq\; \gamma(j,\ell) + 1 + (b_j - \gamma(j,\ell)) = 1 + b_j \leq 2 \cdot b_j,$$

where the last inequality holds since $\min_j b_j \geq 1$. Since $\textit{Dual}_j = b_j$ it follows that $\textit{Primal}_j \leq 2 \cdot \textit{Dual}_j$. After dealing with each request, the primal variables $\{x_e\}_e \cup \{z_i\}_i$ constitute a feasible primal solution. Using weak duality and since $\textit{Primal}_j \leq 2 \cdot \textit{Dual}_j$, it follows that: $B_j^T \cdot Y_j^* \leq X^T \cdot C + Z_j^T \cdot \vec{1} \leq 2 \cdot B_j^T \cdot Y_j$ which proves 2-competitiveness.

We now prove β-feasibility of the dual solution, i.e., for every j, $A_j \cdot Y_j \leq \beta \cdot C$ and $D_j \cdot Y_j \leq \vec{1}$. First we prove the following lemma. Let $\text{row}_e(A)$ denote the eth row of A.

Lemma 1. *For every $j \geq 0$,*

$$x_e^{(j)} \geq \frac{1}{(\max_{i,\ell} w(i,\ell))} \cdot \left(2^{\mathit{row}_e(A_j) \cdot Y_j / c_e} - 1\right).$$

Proof. The proof is by induction on j.

Base $j = 0$: Since the variables are initialized to zero the lemma follows.

Step: The update rule in Step 2c is

$$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot \left(2^{A_{e,(j,\ell)}/c_e} - 1\right).$$

Plugging the induction hypothesis in the update rule implies:

$$\begin{aligned}
x_e^{(j)} &= x_e^{(j-1)} \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot \left(2^{A_{e,(j,\ell)}/c_e} - 1\right) \\
&\geq \frac{1}{(\max_{i,\ell} w(i,\ell))} \cdot \left(2^{\text{row}_e(A_{j-1}) \cdot Y_{j-1}/c_e} - 1\right) \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot \left(2^{A_{e,(j,\ell)}/c_e} - 1\right) \\
&\geq \frac{1}{(\max_{i,\ell} w(i,\ell))} \cdot \left(2^{\text{row}_e(A_j) \cdot Y_j/c_e} - 2^{A_{e,(j,\ell)}/c_e}\right) + \frac{1}{(\max_{i,\ell} w(i,\ell))} \cdot \left(2^{A_{e,(j,\ell)}/c_e} - 1\right) \\
&\geq \frac{1}{(\max_{i,\ell} w(i,\ell))} \cdot 2^{\text{row}_e(A_j) \cdot Y_j/c_e} - \frac{1}{(\max_{i,\ell} w(i,\ell))}.
\end{aligned}$$

The lemma follows. □

Since for every row e of A, $\min_{j,\ell} A_{e,(j,\ell)} \in \{0\} \cup [1, \infty)$, it follows that in Step 2c it holds that for every e such that $A_{e,(j,\ell)} \neq 0$,

$$x_e^{(j)} < b_j \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j,\ell)} \cdot \left(2^{A_{e,(j,\ell)}/c_e} - 1\right).$$

Since for every row e of A, $(\max_{i,\ell} A_{e,(i,\ell)}) \leq c_e$, $\min_{j,\ell} A_{e,(j,\ell)} \in \{0\} \cup [1, \infty)$, and since for every column (i, ℓ) of A, $w(i, \ell) > 0$, and $(\min_i b_i) \geq 1$, it follows that,

$$x_e^{(j)} \leq 2 \cdot b_j + 1 \leq 3 \cdot b_j \ .$$

Lemma 1 implies that:

$$\frac{1}{(\max_{i,\ell} w(i,\ell))} \cdot (2^{\text{row}_e(A_j)\cdot Y_j/c_e} - 1) \quad \leq \quad x_e \leq 3 \cdot b_j \leq 3 \cdot (\max_i b_i) \ .$$

Hence,

$$\text{row}_e(A_j) \cdot Y_j \leq \log_2[1 + 3 \cdot (\max_{i,\ell} w(i,\ell)) \cdot (\max_i b_i)] \cdot c_e \ ,$$

for every j, as required. □

Remark 1. *The assumption in Theorem 2 that* $\max_{j,\ell} A_{e,(j,\ell)} \leq c_e$ *for every row* e *means that the requests are feasible, i.e., do not overload any resource. In our modeling, if* r_j *is infeasible, then* r_j *is rejected upfront (technically,* $\Delta_j = \emptyset$*). Infeasible requests can be scaled to reduce the loads so that the scaled request is feasible. This means that a scaled request is only partially served. In fact, multiple copies of the scaled request may be input (see [7] for a fractional splitting of requests). In addition, in some applications, the oracle procedure is an approximate bi-criteria algorithm, i.e., it finds an embedding that violates capacity constraints. In such a case, we can scale the request to obtain feasibility.*

If a solution Y is (α, β)-competitive, then Y/β is $\alpha \cdot \beta$-competitive. Thus, we conclude with the following corollary.

Corollary 1. *The GVOP algorithm computes a solution* Y *such that* Y/β *is a fractional* $O(\beta)$*-competitive solution.*

Consider the case that the capacities are larger than the demands by a logarithmic factor, namely, $\min_e c_e/\beta \geq \max_{j,\ell} A_{e,(j,\ell)}$ for every row e of A. In this case, we can obtain an all-or-nothing solution if we scale the capacities C in advance as summarized in the following corollary.

Corollary 2. *Assume* $\min_e c_e/\beta \geq \max_{j,\ell} A_{e,(j,\ell)}$*. Run the GVOP algorithm with scaled capacities* C/β*. The solution* Y *is an all-or-nothing* $O(\beta)$*-competitive solution.*

4.3. A Reduction of Requests with Durations

We now add durations to each request. This means each request r_j is characterized, in addition, by a duration interval $T_j = [t_j^{(0)}, t_j^{(1)}]$, where r_j arrives in time $t_j^{(0)}$ and ends in time $t_j^{(1)}$. Requests appear with increasing arrival times, i.e., $t_j^{(0)} < t_{j+1}^{(0)}$. For example, the capacity constraints in virtual circuits now require that, in each time unit, the bandwidth reserved along each edge e is at most c_e. The benefit obtained by serving request r_j is $b_j \cdot |T_j|$, where $|T_j| = t_j^{(1)} - t_j^{(0)}$. We now present a reduction to the general framework.

Let $\tau(j,t)$ denote a 0-1 square diagonal matrix of dimensionality $\sum_{i \leq j} |\Delta_i|$. The diagonal entry corresponding to $f_{i,\ell}$ equals one if and only if request r_i is active in time t, i.e., $\tau(j,t)_{(i,\ell),(i,\ell)} = 1$ iff $t \in T_i$. The capacity constraints are now formulated by

$$\forall t : A_j \cdot \tau(j,t) \cdot Y_j \leq C.$$

Since $\tau(j,t)$ is a diagonal 0-1 matrix, it follows that each entry in $A(j,t) \triangleq A_j \cdot \tau(j,t)$ is either zero or equals the corresponding entry in A_j . Thus, the assumption that $\max_{j,\ell} A_{e,(j,\ell)} \leq c_e$ and that $\min_{j,\ell} A_{e,(j,\ell)} \in \{0\} \cup [1, \infty)$ still hold. This implies that durations of requests simply increase the number of capacity constraints; instead of $A_j \cdot Y_j \leq C$, we have a set of N constraints for every time unit. Let $T_{\max}$ denote $\max_j T_j$. Let $\widetilde{A}_j$ denote the $N \cdot (t_j^{(0)} + T_{\max}) \times \sum_{i \leq j} |\Delta_i|$ matrix obtained by "concatenating" $A(j,1), \ldots, A(j,t_j^{(0)}), \ldots, A(j, t_j^{(0)} + T_{\max})$. The new capacity constraint is simply $\widetilde{A}_j \cdot Y_j \leq C$.

Fortunately, this unbounded increase in the number of capacity constraints has limited implications. All we need is a bound on the "weight" of each column of $\widetilde{A}_j$. Consider a column (i,ℓ) of $\widetilde{A}_j$. The entries of this column are zeros in $A(j,t')$ for $t' \notin T_i$. It follows that the weight of column (i,ℓ) in $\widetilde{A}_j$ equals $|T_i|$ times the weight of column (i,ℓ) in $A(i, t_i^{(0)})$. This implies that the competitive ratio increases to $(2, \beta')$-competitiveness, where $\beta' \triangleq \log_2(1+3 \cdot T_{\max} \cdot (\max_{j,\ell} w(j,\ell)) \cdot (\max_j b_j))$.

Theorem 3. *The GVOP algorithm, when applied to the reduction of online packing with durations, is a* $(2, \beta')$*-competitive online algorithm.*

Remark 2. *Theorem 3 can be extended to competitiveness in* time windows *[5]. This means that we can extend the competitiveness with respect to time intervals* $[0,t]$ *to any time window* $[t_1, t_2]$.

Remark 3. *The reduction of requests with durations to the online packing framework also allows requests with* split intervals *(i.e., a union of intervals). The duration of a request with a split interval is the sum of the lengths of the intervals in the split interval.*

Remark 4. *In the application of circuit switching, when requests have durations, it is reasonable to charge the request "per bit". This means that $b_j/(d_j \cdot |T_j|)$ should be within the range of prices charged per bit. In fact, the framework allows for varying bit costs as a function of the time (e.g., bandwidth is more expense during peak hours). See also [5] for a discussion of benefit scenarios.*

4.4. Approximate Oracles

The GVOP algorithm relies on a VNet embedding "oracle" which computes resource-efficient realizations of the VNets. In general, the virtual network embedding problem is computationally hard, and thus Step 1 could be NP-hard (e.g., a min-cost Steiner tree). Such a solution is useless in practice and hence, we extend our framework to allow for *approximation algorithms* yielding efficient, approximate embeddings. Interestingly, we can show that suboptimal embeddings do not yield a large increase of the competitive ratio as long as the suboptimality is bounded.

Concretely, consider a ρ-approximation ratio of the embedding oracle, i.e., $\gamma(j,\ell) \leq \rho \cdot \min\{\gamma(j,\ell) : f_{j,\ell} \in \Delta_j\}$. The GVOP algorithm with a ρ-approximate oracle requires two modifications: (i) Change the condition in Step 2 to $\gamma(j,\ell) \leq b_j \cdot \rho$. (ii) Change Step 2b to $z_j \leftarrow b_j \cdot \rho - \gamma(j,\ell)/\rho$.

The following theorem summarizes the effect of a ρ-approximate oracle on the competitiveness of the GVOP algorithm.

Theorem 4. *Let $\beta_\rho \triangleq \rho \cdot \log_2(1 + 3 \cdot \rho \cdot (\max_{j,\ell} w(j,\ell)) \cdot (\max_j b_j))$. Under the same assumptions of Theorem 2, the GVOP algorithm is a $(2, \beta_\rho)$-competitive online all-or-nothing packing algorithm if the oracle is ρ-approximate.*

5. Application to VNet Service Models

In this section we show how the framework for online packing can be applied to online VNet embeddings. The key issue that needs to be addressed is the oracles in Line 1 of the GVOP algorithm.

We consider the three traffic models: customer-pipe, hose and aggregate ingress. We also consider three routing models: multipath, single path and tree routing.

Recall that β in Theorem 2 is the factor by which the GVOP algorithm augments resources. Recall that β' is the resource augmentation if VNet requests have durations. The following corollary summarizes our main result as stated in Theorem 1. The following corollary states the values of β and β' when applying Theorems 2 and 3 to the cases described below.

Corollary 3. *The values of β and β' in Theorems 2 and 3 are $\beta = O(\log(|E| \cdot (\max_e c_e) \cdot (\max_j b_j)))$ and $\beta' = O(\log(|T_{\max}| \cdot |E| \cdot (\max_e c_e) \cdot (\max_j b_j)))$ for any sequence of VNet requests from the following types: (i) customer pipe model with multipath routing, (ii) hose model with multipath routing, single path routing[1], or tree routing[1], or (iii) aggregate ingress model with multipath routing, single path routing, or tree routing.*

Remark 5. *Our framework can handle heterogeneous VNet requests, i.e., requests from any of the customer service models and routing models included in Corollary 3. Each time a request arrives, the corresponding oracle procedure is invoked, without disturbing existing requests. This implies that the network resources are shared between requests of all types.*

5.1. Proof of Corollary 3

The proof deals with each traffic model separately.

Customer Pipe Model. In multipath routing, an embedding of a request is a multicommodity flow. The flow along each edge equals the bandwidth reservation needed to support the request. This means that, for each request r_j, the set of valid embeddings Δ_j of r_j consists of all the multicommodity flows specified by the traffic matrix and the edge capacities. For a multicommodity flow $f \in \Delta_j$, the entry $A_{e,f}$ equals the flow $f(e)$. The oracle needs to compute a min-cost multicommodity flow in Δ_j, where a cost of a unit flow along an edge e equals x_e. A min-cost multicommodity flow can be computed by solving a linear program or by a using a PTAS [33].

A technical issue that needs to be addressed is that the flow along an edge may be positive yet smaller than one, thus violating the requirement in Theorem 2. A key observation is that the requirement in Theorem 2 can be relaxed to $A_{e,(j,\ell)} \in \{0\} \cup [\frac{1}{N^2}, \infty)$. This only affects the augmentation by a constant factor. Thus, one can deal with this issue by peeling off such a flow, and rerouting it along

[1]The oracle in this case does not run in polynomial time.

other paths. On the other hand, the resulting oracle in this case may violate edge capacities. We refer the reader to [19] where an extension of GVOP that deals with such an oracle is discussed in detail.

Hose Model. In multipath routing, an embedding is a reservation u of bandwidths to edges so that every allowed traffic can be routed as a multicommodity flow. An entry $A_{e,(j,u)}$ equals the bandwidth u_e reserved in e for the embedding of request r_j. In [18], a linear programming based poly-time algorithm is presented for a min-cost reservation in the hose model. As in the case of the customer pipe model, positive bandwidth reservations of an edge might be smaller than 1. The oracle in this case executes the algorithm in [18] to obtain an optimal reservation. This reservation is modified so that (1) the minimum positive bandwidth reservation is $\Omega(1/m^2)$, (2) edge capacities are violated at most by a constant factor, and (3) the cost of the embedding is at most doubled. We refer the reader to [19] where an extension of GVOP that deals with such an oracle is discussed.

An efficient approximate oracle for tree routing in the hose model is an open problem [29]. We elaborate on a non-polynomial oracle that focuses on the online aspects of the problem. A Steiner tree T is feasible if and only if the bandwidth $u(e)$ reserved for each edge e equals the maximum traffic that may traverse e and $u(e) \leq c_e$. Indeed, let $A_e \cup B_e$ denote a partitioning of the terminals U_j induced by the deletion of the edge e from T. The maximum traffic along e by request r_j equals

$$\min\left\{\sum_{u\in A_e} b_{\mathit{out}}(u), \sum_{v\in B_e} b_{\mathit{in}}(v)\right\} + \min\left\{n\sum_{u\in A_e} b_{\mathit{in}}(u), \sum_{v\in B_e} b_{\mathit{out}}(v)\right\}.$$

The non-polynomial oracle simply returns a min-cost feasible Steiner tree that spans the set of terminals U_j. If no feasible Steiner tree exists, the request is rejected. A similar non-polynomial oracle exists for single path routing.

Aggregate Ingress Model. An embedding in the aggregate ingress model is also a reservation of bandwidths so that every allowed traffic can be routed. In the multipath routing model, a min-cost embedding can be obtained by a variation of the algorithm presented in [18] combined with the a modification that avoids small reservations as discussed previously for the customer pipe and hose models. Dealing with small reservations is done similarly to the way described in the hose model case.

A min-cost single path routing embedding in the aggregate ingress model is always a tree with uniform bandwidth reservation that equals the ingress amount.

Thus, the routing models of single paths and trees coincide in this case. This implies that a min-cost tree embedding is simply a min-cost Steiner tree. The oracle in this case proceeds as follows. (1) Delete all edges the capacity of which is less than the ingress amount. If this deletion disconnects the terminals in U_j, then reject the request. (2) Compute a min-cost Steiner tree over the remaining edges (see [32] and [13] for the best approximation to date).

5.2. Router Loads

So far we have focused on the load incurred over the edges, i.e., the flow (e.g., data rate) along an edge is bounded by the edge capacity (e.g., available bandwidth). In this section we also view the nodes of the network as resources. We model the load incurred over the nodes by the rate of the packets that traverse a node. Thus, a request is characterized, in addition, by the so-called *packet rate*.

In this setting, each node (router) v has a computational capacity c_v that specifies the maximum rate of packets that node v can process. The justification for modeling the load over a node in this way is that a router must inspect each packet. The capacity constraint of a node v simply states that the sum of the packet rates along edges incident to v must be bounded by c_v.

For simplicity, we consider the aggregate ingress model with tree routing. A request r_j has an additional parameter pr_j that specifies the aggregate ingress packet rate, i.e., pr_j is an upper bound on the sum of the packet rates of all ingress traffic for request r_j.

Applying our framework requires to add a row in A to each node (in addition to a row per edge). An entry $A_{v,u}$ equals pr_j if the reservation u of capacities assigns a positive capacity to an edge incident to v, and zero otherwise. The oracle now needs to compute a node-weighted Steiner tree [27]. The approximation ratio for this problem is $O(\log k_j)$, where k_j denotes the number of terminals in request r_j.

The following corollary summarizes the values of ρ and β_ρ when applying Theorem 4 to router loads. One can extend also Theorem 3 in a similar fashion.

Corollary 4. *In the aggregate ingress model with tree routing,* $\rho = O(\log \max_j k_j)$ *and* $\beta_\rho = O(\rho \cdot \log(\rho \cdot (|E| \cdot (\max_e c_e) + |V| \cdot (\max_v c_v)) \cdot (\max_j b_j)))$.

6. Discussion

This article presents a unified algorithm for online embeddings of VNets. Each VNet request consists of endpoints, quality-of-service constraints, and a routing model. The algorithm can handle VNets requests from different models (e.g., the customer-pipe, hose, and aggregate-ingress models), and each request may allow a different routing model (e.g., multipath, single-path, and tree-routing). Since the problem we address is a generalization of online circuit switching [5], it follows that the lower bounds apply to our case as well. Namely, the competitive ratio of any online algorithm is $\Omega(\log(n \cdot T_{\max}))$, where n denotes the number of nodes and $T_{\max}$ is the maximal duration.

In the context of hose model with tree routing, we emphasize that finding a feasible Steiner tree is NP-hard to approximate within a constant factor [29]. One can relax the feasibility requirement, and compute a Steiner tree that violates the capacities of G by a factor of $\mu \geq 1$ while satisfying the ingress/egress demand of every terminal. The oracle in this case will be bi-criteria. Bi-criteria oracles can be incorporated in the primal-dual scheme as shown in [19]. To our knowledge, the question whether there is a polynomial time bi-criteria algorithm for min-cost Steiner trees in the hose model is open.

Acknowledgments

Part of this work was performed within the Virtu project, funded by NTT DoCoMo Euro Labs, and the Collaborative Networking project, funded by Deutsche Telekom AG. We would like to thank our colleagues in these projects for many fruitful discussions. We are grateful to Ernesto Abarca and Johannes Grassler for their help with the prototype architecture [31], and to Boaz Patt-Shamir for initial discussions.

References

[1] R. Ahlswede, N. Cai, S. Li, and R. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[2] D. Andersen. Theoretical approaches to node assignment. In *http://www.cs.cmu.edu/ dga/papers/andersenassignabstract.html.*, 2002.

[3] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts. On-line routing of virtual circuits with applications to load balancing and machine scheduling. *Journal of the ACM (JACM)*, 44(3):486–504, 1997.

[4] B. Awerbuch and Y. Azar. Competitive multicast routing. *Wirel. Netw.*, 1, 1995.

[5] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive on-line routing. In *Proc. IEEE FOCS*, 1993.

[6] B. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts. Competitive routing of virtual circuits with unknown duration. *Journal of Computer and System Sciences*, 62(3):385–397, 2001.

[7] Y. Azar and R. Zachut. Packet routing and information gathering in lines, rings and trees. In *Proc. ESA*, pages 484–495, 2005.

[8] N. Bansal, K.-W. Lee, V. Nagarajan, and M. Zafer. Minimum congestion mapping in a cloud. In *Proc. ACM PODC*, pages 267–276, 2011.

[9] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, New York, NY, USA, 1998.

[10] N. Buchbinder and J. S. Naor. Improved bounds for online routing and packing via a primal-dual approach. *Proc. IEEE FOCS*, 2006.

[11] N. Buchbinder and J. S. Naor. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science*, 3(2-3):99–263, 2009.

[12] N. Buchbinder and J. S. Naor. Online primal-dual algorithms for covering and packing. *Math. Oper. Res.*, 34(2):270–286, 2009.

[13] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. An improved LP-based approximation for Steiner tree. In *Proc. ACM STOC*, pages 583–592, 2010.

[14] C. Chekuri, F. B. Shepherd, G. Oriolo, and M. G. Scutellá. Hardness of robust network design. *Netw.*, 50(1):50–54, 2007.

[15] N. Chowdhury and R. Boutaba. A survey of network virtualization. *Computer Networks*, 2009.

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

[17] F. Eisenbrand and F. Grandoni. An improved approximation algorithm for virtual private network design. In *Proc. ACM SODA*, 2005.

[18] T. Erlebach and M. Ruegg. Optimal bandwidth reservation in hose-model VPNs with multi-path routing. In *Proc. IEEE INFOCOM*, pages 2275–2282, 2004.

[19] G. Even and M. Medina. Online multi-commodity flow with high demands. Manuscript (see: `www.eng.tau.ac.il/~medinamo`).

[20] J. Fan and M. H. Ammar. Dynamic topology configuration in service overlay networks: A study of reconfiguration policies. In *Proc. IEEE INFOCOM*, 2006.

[21] J. Fingerhut, S. Suri, and J. Turner. Designing least-cost nonblocking broadband networks. *J. Algorithms*, 24(2):287–309, 1997.

[22] F. Grandoni and T. Rothvoss. Network design via core detouring for problems without a core. In *Proc. ICALP*, 2010.

[23] K. Grewal and S. Budhiraja. Performance evaluation of on-line hose model VPN provisioning algorithm. *Advances in Computer Vision and Information Technology*, 2008.

[24] A. Gupta, A. Kumar, and T. Roughgarden. Simpler and better approximation algorithms for network design. In *Proc. ACM STOC*, pages 365–372, 2003.

[25] G. Italiano, S. Leonardi, and G. Oriolo. Design of trees in the hose model: the balanced case. *Operations Research Letters*, 34(6):601–606, 2006.

[26] A. Juttner, I. Szabo, and A. Szentesi. On bandwidth efficiency of the hose resource management model in virtual private networks. In *Proc. IEEE INFOCOM*, 2003.

[27] P. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted Steiner trees. *J. Algorithms*, 19(1):104–115, 1995.

[28] M. Kodialam, T. Lakshman, and S. Sengupta. Online multicast routing with bandwidth guarantees: a new approach using multicast network flow. *IEEE/ACM Transactions on Networking (TON)*, 11(4):676–686, 2003.

[29] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener. Algorithms for provisioning virtual private networks in the hose model. *IEEE/ACM Trans. Netw.*, 10(4), 2002.

[30] Y. Liu, Y. Sun, and M. Chen. MTRA: An on-line hose-model VPN provisioning algorithm. *Telecommunication Systems*, 31(4):379–398, 2006.

[31] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy. Network virtualization architecture: Proposal and initial prototype. In *Proc. ACM VISA*, pages 63–72. ACM, August 2009.

[32] V. V. Vazirani. Recent results on approximating the Steiner tree problem and its generalizations. *Theor. Comput. Sci.*, 235(1):205–216, 2000.

[33] N. Young. Sequential and parallel algorithms for mixed packing and covering. In *Proc. 42nd IEEE FOCS*, 2001.

[34] Y. Zhu and M. H. Ammar. Algorithms for assigning substrate network resources to virtual network components. In *Proc. IEEE INFOCOM*, 2006.