Competitive and Deterministic Embeddings of Virtual Networks

Guy Even¹, Moti Medina¹, Gregor Schaffrath², Stefan Schmid²

¹ School of Electrical Engineering Tel-Aviv Univ. Tel-Aviv 69978, Israel.
{guy,medinamo}@eng.tau.ac.il
² Deutsche Telekom Labs, TU Berlin, 10587 Berlin, Germany.
{grsch,stefan}@net.t-labs.tu-berlin.de

Abstract

Network virtualization is a paradigm that allows for flexible and efficient allocation of the resources among multiple virtual networks (VNets). In this paper we deal with the problem of embedding dynamically arriving VNet requests. We describe a generic algorithm for the online VNet embedding problem and analyze its competitive ratio. This means that we compare the benefit accumulated by the algorithm with the benefit of an optimal offline algorithm. We prove that the competitive ratio of our online algorithm 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 allows for nonuniform benefits and durations of VNet requests. Concretely, the routing models considered in this paper include: multipaths, single paths, and tree routing. For modeling traffic, we study the customer-pipe model, the hose model, and a new traffic model, called aggregate ingress model, that is well suited for modeling multicasts and multi-party video conferences.

Keywords: Network virtualization, online algorithms, resource allocation.
1 Introduction

Virtualization is a method for providing an abstraction of a logical entity over a physical one. Classic examples of virtualization include: virtual memory, partitioning of a hard disk into logical disks, and running multiple operation systems on the same computer. Recently, virtualization offered by data-centers and cloud-computing has changed the possibilities for storage and computing. There are many advantages that justify virtualization, among them [29]: (i) clients avoid capital expenses and acquisition of expertise needed to build the infrastructure, (ii) a client is charged according to usage, thus relieving the need to accurately forecast demand, (iii) cheaper infrastructure thanks to economy-of-scale, and (iv) improved reliability. Network virtualization over the Internet offers the additional advantages of guaranteeing quality-of-service and encapsulating an abstraction free from the strict limitations of the Internet [13]. This encapsulation also enables testbeds in which innovation can be safely experimented with [3].

The basic entity offered by network virtualization is a virtual network (VNet), that is, an abstraction of a network over a given subset of terminals. In general, establishing a VNet requires embedding a subnet-work that spans the terminals of the VNet in the physical network (often called the substrate network). A main characteristic of a VNet is a specification of allowable traffic between terminals, and the embedding associated with a VNet must support this specification. The incentive to embed a request for a VNet is that a benefit is associated with each request.

A central challenge in network virtualization is the efficient use of the given physical resources, i.e., how to satisfy as many VNet requests as possible with the given resources. Efficient utilization of resources allows also for reduced power consumption or shifting of tasks to locations where power is available. As the requests typically arrive over time and are hard to predict, the decisions on how to use the resources must be made with limited knowledge.

This paper deals with the question of how to embed VNets arriving one-by-one in an online fashion [7]. Each request either needs to be embedded or rejected. The online setting means that the decision (embed or reject) must be taken without any information about future requests, and this decision cannot be changed later.

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 $\alpha$ if, for every sequence of requests $\sigma$, the benefit obtained by the algorithm is at least an $\alpha$ fraction of the optimal offline benefit obtainable for $\sigma$.

1.1 VNet Service Models

There are many service models for VNets [24], 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. (2) In the hose model [14] [18], each terminal $v$ is assigned a maximum ingress bandwidth $b_{in}(v)$ and a maximum egress bandwidth $b_{out}(v)$. 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 $I$. Any traffic in which the sum of ingress bandwidths is at most $I$ 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 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 [16, 15, 19, 21].

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 $I$: (i) a single multicast from one node with bandwidth $I$, and (ii) a set of multicast sessions with bandwidths $f_i$, where $\sum_i f_i \leq I$. 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 video from the other participants.

**Routing.** We briefly outline three models for the allowed routing. (1) **Tree routing**, the VNet is embedded as a Steiner tree in the substrate network that spans the terminals of the VNet. (2) **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) **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 bandwidth as the main resource of a link. However, throughput can also depend on the capacity 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.

**Duration and Benefit.** The algorithms presented in this paper 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 [13]. A description of a prototype network virtualization architecture under development in Deutsche Telekom Labs appears in [30].

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] [12]). There exist several results for the offline variant of the embedding problem. In the customer-pipe model, an optimal multipath
fractional solution is obtained by solving a multicommodity flow problem. An integral reservation for multipath routing is equivalent to the generalized Steiner network problem for which a 2-approximation is known [23]. In the hose model, constant approximation algorithms have been developed for tree routing [15, 19, 21]. Moreover, the cost of the tree competes with the optimal single path routing. 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 [22] (see also [27]). Finally, an optimal reservation for multipath routing in the hose model is presented in [16].

Published online algorithms for VNet embeddings are scarce. In [20, 28], 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, using such linear edge costs leads to trivial linear competitive ratios [5]. The rejection ratio of the algorithm is analyzed in [20, 28], but not the competitive ratio. The problem of embedding multicast requests in an online setting was studied in [26]. They used a heuristic oracle that computes a directed Steiner tree. The competitive ratio of the algorithm in [26] is not studied. In fact, much research has focused on heuristic approaches, e.g., [17] proposes heuristic methods for constructing different flavors of reconfiguration policies; and [33] 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.

Circuit switching can be regarded as a special case of VNet embeddings as each VNet consists of two terminals. Online algorithms for circuit switching were presented in [5]. A general primal-dual setting for online packing and covering appears in [8, 10].

1.3 Our Contribution

This paper describes an algorithmic framework called General Integral Packing Online Algorithm (GIPO) for online embeddings of VNet requests. It follows the primal-dual online packing scheme by Buchbinder and Naor [8, 10] that also provides an explanation of the algorithm of Awerbuch et al. [5].

In our eyes, the main contribution of this paper lies in the generality of the algorithm in terms of supported traffic and routing models. In particular, we introduce a traffic model, called aggregate ingress model, that allows a router to duplicate packets to support efficient multicasting and broadcasting. In the aggregate ingress model, the set of allowable traffic patterns is simply specified by the set of terminals and the sum of ingress rates of the terminals. The aggregate ingress model is well suited for uniformly modeling unicasts, multicasts, and broadcasts and supports efficient multicasting and broadcast.

In summary, the algorithm presented in this paper allows the VNet requests to follow the important customer-pipe models, hose models, or aggregate ingress models, and routing can either be multipath, single path, or on trees. Thus, different requests may belong to different traffic and routing types. This implies that the network resources can be fully shared between requests of all types.

We prove that the competitive ratio of our online algorithm is, in essence, logarithmic in the resources of the network. The algorithm is deterministic and centralized. The algorithm has two versions: (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 request’s benefit. 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 Paper Organization

The remainder of this paper is organized as follows. We introduce the formal model and problem in Section 2. Subsequently, the algorithmic framework (Section 4) is described. 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 extension to embeddings where also the router load (given, e.g., by the packet rates), has to be taken into account is investigated in Section 6. The paper concludes with a short discussion in Section 7.

2 Problem Definition

We assume an undirected communication network $G = (V, E)$ where $V$ represents the set of (substrate) nodes 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. Edges are associated with capacities (e.g., bandwidth), i.e., $c : E \rightarrow \mathbb{R}_{\geq 0}$ denotes the link capacity function. In Section 6, we will extend the model also to node capacities (i.e., processing power of a node).

The operator of the communication 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. (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 capacity reservations that induces a tree that spans the terminals. The reserved capacities must not exceed the edge capacities. In addition, the traffic must be routable in the tree with the reserved capacities.

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 may consume a larger fraction of the resources, then the operator can accept and embed a fraction of a request. If an operator accepts an $\epsilon$-fraction of $r_j$, then this means that it uniformly serves an $\epsilon$-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 $\epsilon$-fraction of $r_j$ is $\epsilon \cdot b_j$. The goal is to maximize the sum of the received benefits.

3 The Main Result

We define below what a bi-criteria online algorithm is. 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.

4
**Definition 1.** Let $OPT$ denote an optimal offline fractional packing solution. An online packing algorithm $Alg$ is $(\alpha, \beta)$-competitive if: (i) For every input sequence $\sigma$, the benefit of $Alg(\sigma)$ is at least $\frac{1}{\alpha}$ times the benefit of $OPT$. (ii) For every input sequence $\sigma$ and for every resource $e$, the congestion incurred by $Alg(\sigma)$ is at most $\beta$.

Our online algorithm GIPO is described in Section 4.2. The main result of this paper 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, the GIPO algorithm is a $(2, \beta)$-competitive online integral VNet embedding algorithm.

The proof of Theorem 1 appears in Sections 4, 5, 6.

### 4 A Framework for Online Packing

Our embedding framework is an adaptation of the online primal-dual framework by Buchbinder and Naor [9, 10]. 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

Our setting can be motivated with the online circuit switching problem from [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 integral or “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.

We now present the linear programming formulation of online packing. To simplify reading, we use the terminology of the online circuit switching problem with durations. Let $\Delta_j$ denote the set of valid embeddings of $r_j$ (e.g., $\Delta_j$ 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 the cardinality of $\Delta_j$ is typically exponential; this implies that the linear programs would be of exponential size. Obviously, an application of the framework does not require an explicit representation (see Section 5).

We formulate the online packing as 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 request $r_1, \ldots, r_j$). Let $B_j$ denote the benefits column vector $(b_1, \ldots, b_j)^T$. 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 $N \times |\Delta_j|$. An entry $(A_j)_{\ell, (i, \ell)}$ 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 \( (D_j)_{i,i'} = 1 \) 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 \( |\Delta_j| \) 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 \( x_e \) 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

The listing of the general packing algorithm appears in Algorithm 1. We assume that all the variables (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 \( \text{col}((j,\ell)) \) denote the column of \( A \) (in fact, \( A_j \)) that corresponds to the dual variable \( y_{j,\ell} \). Let \( \gamma(j,\ell) \triangleq X^T \cdot \text{col}((j,\ell)) \). It is useful to interpret \( \gamma(j,\ell) \) as the \( X \)-cost of the “flow” \( f_{j,\ell} \) for request \( j \). Let \( w(j,\ell) \triangleq \frac{1}{\alpha} X^T \cdot \text{col}((j,\ell)) \), namely, \( w(j,\ell) \) is the sum of the entries in columns \((j,\ell)\) 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 \((\alpha,\beta)\)-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 \( D_j \cdot Y_j \leq \bar{I} \).

The following theorem can be proved using the same techniques as in [2].

**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 \( A \), \( 1 \leq \min_{j,\ell} A_{e,(j,\ell)} \), and (iii) \( \min_{j,\ell} b_j \geq 1 \). Let \( \beta \triangleq \log_2(1 + 3 \cdot (\max_{j,\ell} w(j,\ell)) \cdot (\max_{j} b_j)) \). The GIPO algorithm is a \((2,\beta)\)-competitive online integral packing algorithm.

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

We show that \( \text{Primal}_j \leq 2 \cdot \text{Dual}_j \) for every \( j \). We show that GIPO produces feasible primal solutions throughout its execution. Initially, the primal and the dual solutions are 0, and the claim holds. Let \( x^{(j)}_e \) denote the value of the primal variable \( x_e \) when \( r_j \) is processed. If \( r_j \) is rejected then \( \text{Primal}_j = \text{Dual}_j = 0 \) and the claim holds. Then for each accepted request \( r_j \), \( \text{Dual}_j = b_j \) and \( \text{Primal}_j = \sum_{e \in E(j,\ell)} (x^{(j)}_e - x^{(i-1)}_e) \cdot c_e + z_i \), where \( E(j,\ell) = \{ e \in \{1, \ldots, N \} : A_{e,(j,\ell)} \neq 0 \} \). Step (2b) increases the cost \( X^T \cdot C = \sum_e x_e \cdot c_e \).
Algorithm 1 The General Integral (all-or-nothing) Packing Online Algorithm (GIPO).

Upon the $j$th round:

1. $f_{j,t} \leftarrow \arg\min\{\gamma(j, \ell) : f_{j,t} \in \Delta_j\}$ (oracle procedure)

2. If $\gamma(j, \ell) < b_j$ then, (accept)
   
   (a) $y_{j,t} \leftarrow 1$.
   
   (b) For each row $e$ : If $A_{e,(j,\ell)} \neq 0$ do
      
      $$x_e \leftarrow x_e \cdot 2^{A_{e,(j,\ell)}/c_e} + \frac{1}{w(j, \ell)} \cdot (2^{A_{e,(j,\ell)}/c_e} - 1).$$

   (c) $z_j \leftarrow b_j - \gamma(j, \ell)$.

3. Else, (reject)
   
   (a) $z_j \leftarrow 0$.

as follows:

$$\sum_{e \in E(j,\ell)} (x_e^{(i)} - x_e^{(i-1)}) \cdot c_e \leq \sum_{e \in E(j,\ell)} \left[ x_e \cdot \left(2^{A_{e,(j,\ell)}/c_e} - 1\right) + \frac{1}{w(j, \ell)} \cdot \left(2^{A_{e,(j,\ell)}/c_e} - 1\right) \right] \cdot c_e$$

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

$$\leq \sum_{e \in E(j,\ell)} \left( x_e + \frac{1}{w(j, \ell)} \right) \cdot A_{e,(j,\ell)}$$

$$= \gamma(j, \ell) + 1.$$  

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

$$Primal_j \leq \gamma(j, \ell) + 1 + (b_j - \gamma(j, \ell))$$

$$= 1 + b_j \leq 2 \cdot b_j.$$  

where the last inequality holds since $\min_j b_j \geq 1$. Since $Dual_j = b_j$ it follows that $Primal_j \leq 2 \cdot Dual_j$.

After dealing with each request, the primal variables $\{x_e\} \cup \{z_i\}$ constitute a feasible primal solution.

Using weak duality and since $Primal_j \leq 2 \cdot Dual_j$, it follows that:

$$B_j^T \cdot Y_j^* \leq X^T \cdot C + Z_j^T \cdot \bar{1} \leq 2 \cdot B_j^T \cdot Y_j,$$

which proves $2$-competitiveness.

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

Lemma 3. $x_e \geq \frac{1}{\max_i w(i,\ell)} \cdot (2^{\max_i w(i,\ell)} - 1) \cdot \frac{\sum_{i \cap (j,\ell)} A_j \cdot Y_j}{c_e}$

Proof: The proof is by induction. Base $i = 0$: Since the variables are initialized to zero the lemma follows.

Step: The update rule in Step (2b) is $x_e \leftarrow x_e \cdot 2^{A_e,(j,\ell)/c_e} + \frac{1}{w(j, \ell)} \cdot (2^{A_e,(j,\ell)/c_e} - 1)$. Plugging the induction hypothesis in the update rule implies:
\[ x_e = x_e \cdot 2^{A_{e,(j,\ell)}/e} + \frac{1}{w(j, \ell)} \cdot (2^{A_{e,(j,\ell)}/e} - 1) \]

\[ \geq \frac{1}{(\max_{i,\ell} w(i, \ell))} \cdot (2^{\text{row}(A_{j-1}) \cdot Y_{j-1}/e} - 1) \cdot 2^{A_{e,(j,\ell)}/e} + \frac{1}{w(j, \ell)} \cdot (2^{A_{e,(j,\ell)}/e} - 1) \]

\[ \geq \frac{1}{(\max_{i,\ell} w(i, \ell))} \cdot (2^{\text{row}(A_j) \cdot Y_j/e} - 2^{A_{e,(j,\ell)}/e}) + \frac{1}{(\max_{i,\ell} w(i, \ell))} \cdot (2^{A_{e,(j,\ell)}/e} - 1) \]

\[ \geq \frac{1}{(\max_{i,\ell} w(i, \ell))} \cdot 2^{\text{row}(A_j) \cdot Y_j/e} - \frac{1}{(\max_{i,\ell} w(i, \ell))} \cdot 2^{A_{e,(j,\ell)}/e} \]

The lemma follows. \(\square\)

Since \((\max_{i,\ell} A_{e,(i,\ell)}) \leq c_e, 1 \leq (\min_{i,\ell} A_{e,(i,\ell)}), (\min_{i} b_i) \geq 1\). Step (2b) in the GIPO implies that for every \(e\), \(x_e < b_j \cdot 2^{A_{e,(j,\ell)}/e} + \frac{1}{w(j, \ell)} \cdot (2^{A_{e,(j,\ell)}/e} - 1)\), and hence it holds that for every \(j\), \(x_e \leq 2 \cdot b_j + 1 \leq 3 \cdot b_j\). Lemma 3 implies that:

\[ \frac{1}{(\max_{i,\ell} w(i, \ell))} \cdot (2^{\text{row}(A_j) \cdot Y_j/e} - 1) \leq x_e \leq 3 \cdot b_j \leq 3 \cdot (\max_{i} b_i) \]

Implying that

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

as required. \(\square\)

**Remark 1.** The assumption in Theorem 2 that \(\max_{i,\ell} A_{e,(i,\ell)} \leq c_e\) 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 [6] 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 \((\alpha, \beta)\)-competitive, then \(Y/\beta\) is \(\alpha \cdot \beta\)-competitive. Thus, we conclude with the following corollary.

**Corollary 4.** The GIPO algorithm computes a solution \(Y\) such that \(Y/\beta\) is a fractional \(O(\beta)\)-competitive solution.

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

**Corollary 5.** Assume \(\min_{e} c_e/\beta \geq \max_{i,\ell} A_{e,(j,\ell)}\). Run the GIPO algorithm with scaled capacities \(C/\beta\). 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 \( \sum_{t \leq j} |\Delta_t| \). The diagonal entry corresponding to \( f_{j,\ell} \) equals one if and only if request \( r_i \) is active in time \( t \), i.e., \( \tau(j,t)(i,\ell,\ell) = 1 \) 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 \( \max_{j,\ell} A_{e,(j,\ell)} \leq c_e \) still holds. 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 \( \tilde{A}_j \) denote the \( N \cdot (t_j^{(0)} + T_{\text{max}}) \times \sum_j |\Delta_j| \) matrix obtained by “concatenating” \( A(j,1), \ldots, A(j,t) \). The new capacity constraint is simply \( \tilde{A}_j \cdot Y_j \leq C \).

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 \( \tilde{A}_{j,t} \). Consider a column \( (i, \ell) \) of \( \tilde{A}_{j,t} \). The entries of this column are zeros in \( A(j,t') \) for \( t' \notin T_i \). It follows that the weight of column \( (i, \ell) \) in \( \tilde{A}_{j,t} \) equals \( |T_i| \) times the weight of column \( (i, \ell) \) in \( A(i,t_i^{(0)}) \). This implies that the competitive ratio increases to \( (2, \beta')\)-competitiveness, where \( \beta' \triangleq \log_2(1 + 3 \cdot T_{\text{max}} \cdot \max_{j,\ell} w(j,\ell) \cdot (\max_j b_j)) \).

**Theorem 6.** The GIPO algorithm, when applied to the reduction of online packing with durations, is a \((2, \beta')\)-competitive online algorithm.

**Remark 2.** Theorem 6 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 \( b_j/(\sum_{j} |T_j|) \) 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 expensive during peak hours). See also \([5]\) for a discussion of benefit scenarios.

### 4.4 Approximate Oracles

In our applications the oracle in Step 1 has to solve an NP-hard problem (e.g., a min-cost Steiner tree). Hence, we extend our framework to allow for approximation algorithms yielding efficient, approximate embedding solutions. 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 \( \rho \)-approximation ratio of the embedding oracle, i.e., \( \gamma(j,\ell) \leq \rho \cdot \arg\min \{ \gamma(j, \ell) : f_{j,\ell} \in \Delta_j \} \). The GIPO algorithm with a \( \rho \)-approximate oracle requires two modifications: (i) Change the condition in Step 2 to \( \gamma(j,\ell) \leq b_j \cdot \rho \). (ii) Change Step 3 to \( z_j \leftarrow b_j \cdot \rho - \gamma(j,\ell) \).

The following theorem summarizes the effect of a \( \rho \)-approximate oracle on the competitiveness of the GIPO algorithm.

**Theorem 7.** Let \( \beta_\rho \triangleq \log_2(1 + 3 \cdot \rho \cdot (\max_{j,\ell} w(j,\ell)) \cdot (\max_j b_j)) \). Under the same assumptions of Theorem 6, the GIPO algorithm is a \((1 + \rho, \beta_\rho)\)-competitive online integral packing algorithm if the oracle is \( \rho \)-approximate.
5 Application to Online VNet Embeddings

In this section we show how the framework for online packing can be applied to online VNet embeddings. Since the linear programs have exponential size, explicit representations must be avoided. We consider the three important traffic models customer-pipe, hose and aggregate ingress, and the three main routing models multipath, single path and tree routing. The embeddings in this section focus on edge capacity constraints; in Section 6 we extend the results to router loads.

Recall that $\beta$ in Theorem 2 is the factor by which the GIPO algorithm augments resources. Recall that $\beta'$ is the resource augmentation if VNet requests have durations. The following corollary states the values of $\beta$ and $\beta'$ when applying Theorems 2 and 6 to the cases described below.

**Corollary 8.** The values of $\beta$ and $\beta'$ in Theorems 2 and 6 are

$$\beta = O\left(\log(|E| \cdot (\max_e c_e \cdot (\max_j b_j)))\right)$$

and

$$\beta' = O\left(\log(|T_{\text{max}}| \cdot |E| \cdot (\max_e c_e \cdot (\max_j b_j)))\right)$$

for any sequence of VNet requests from 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.

**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 8. Each time a request arrives, the corresponding oracle procedure is invoked, without disturbing existing requests. This implies that the network resources can be fully shared between requests of all types.

5.1 Customer Pipe Model

In multipath routing, an embedding of a request is a multicommodity flow. This means that, for each request $r_j$, the set of valid embeddings $\Delta_j$ 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 $\Delta_j$, 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 [32].

5.2 Hose Model

In multipath routing, an embedding is a reservation $u$ of capacities so that every allowed traffic can be routed as a multicommodity flow. An entry $A_{e,u}$ equals the capacity $u_e$ reserved in $e$ for the embedding of request $r_j$. In [16], a linear programming based polytime algorithm is presented for a min-cost reservation in the hose model.

In [15, 19, 21] constant approximation ratio algorithms are presented for min-cost reservations in the hose model. These algorithms return a tree routing whose cost is at most a constant factor larger than the cost of an optimal single path routing. This implies that we can employ tree routing (which is easier to manage) and compete with single path routing (which is harder to manage but supposedly cheaper).

5.3 Aggregate Ingress Model

An embedding in the aggregate ingress model is also a reservation of capacities so that every allowed traffic can be routed. In the multipath routing model, an optimal linear programming based polytime algorithm for a min-cost embedding can be obtained by a variation of the algorithm presented in [16].

A min-cost single path routing embedding in the aggregate ingress model is always a tree. Thus, the routing models of single paths and trees coincide. Moreover, the reservation along every edge equals the
aggregate ingress $I$. This implies that a min-cost tree embedding is simply a min-cost Steiner tree. Many constant approximation algorithms for min-cost Steiner trees have been published [31], the best result to date is [11].

6 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 capacity reservation $u$ assigns positive capacity to an edge incident to $v$, and zero otherwise. The oracle now needs to compute a node-weighted Steiner tree [25]. 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 $\rho$ and $\beta_\rho$ when applying Theorem 7 to router loads. One can extend also Theorem 6 in a similar fashion.

**Corollary 9.** In the aggregate ingress model with tree routing, $\rho = O(\log \max_j k_j)$ and $\beta_\rho = O(\log(\rho \cdot (|E| \cdot (\max_e c_e) + |V| \cdot (\max_v c_v)) \cdot (\max_j b_j)))$.

7 Discussion

We present a unified algorithm for online embeddings of VNets. The algorithm handles VNets requests in several important models (namely, the customer-pipe, hose, and aggregate-ingress models), and each request may allow multipath/single-path/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.

The all-or-nothing version of our algorithm violates the capacity constraints if the capacities are small compared to the demands. Such a violation can be viewed as resource augmentation or “over-booking” and justified by stochastic behavior of demand. Namely, there is a gap between the sum of the peak demands of the requests and the peak demand of the sum of the requests.

We believe that our work opens the doors to many fundamental questions in the field of network virtualization and can, e.g., give advice on how to optimally share a given infrastructure. Moreover, VNet embedding strategies are closely related to many important economical questions such as VNet pricing: given the benefit obtained from embedding a VNet and the corresponding embedding cost (in terms of resources), which requests are worthwhile to accept, and what is the value of a resource?

Acknowledgments

Part of this work was performed within the 4WARD project, which is funded by the European Union in the 7th Framework Programme (FP7), 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: Anja Feldmann, Dan Jurca, Wolfgang Kellerer, Ashiq Khan, Kazuyuki Kozu, and Joerg Widmer. Special thanks also go to Ernesto Abarca for the great help during the implementation of our prototype network virtualization architecture \[30\]. Finally, Stefan Schmid would like to thank Boaz Patt-Shamir from Tel Aviv University for initial discussions on the subject.

References


