It's About Time: On Optimal Virtual Network Embeddings under Temporal Flexibilities

Matthias Rost*, Stefan Schmid*†, Anja Feldmann*
*Internet Network Architectures, Technische Universität Berlin, Germany
†Telekom Innovation Laboratories, Germany

***Abstract*—Distributed applications often require high-performance networks with strict connectivity guarantees. For instance, many cloud applications suffer from today's variations of the intra-cloud bandwidth, which leads to poor and unpredictable application performance. Accordingly, we witness a trend towards virtual networks (VNets) which can provide resource isolation. Interestingly, while the problem of *where* to embed a VNet is fairly well-understood today, much less is known about *when* to optimally allocate a VNet. This however is important, as the requirements specified for a VNet do not have to be static, but can vary over time and even include certain temporal flexibilities.**

This paper initiates the study of the *temporal VNet embedding problem* (TVNEP).We propose a *continuous-time* mathematical programming approach to solve the TVNEP, and present and compare different algorithms. Based on these insights, we present the cΣ-Model which incorporates both symmetry and state-space reductions to significantly speed up the process of computing exact solutions to the TVNEP. Based on the cΣ-Model, we derive a greedy algorithm $c\Sigma_A^G$ to compute fast approximate solutions.

In an extensive computational evaluation, we show that despite the hardness of the TVNEP, the cΣ-Model is sufficiently powerful to solve moderately sized instances to optimality *within one hour*, and under different *objective functions* (such as maximizing the number of embeddable VNets). We also show that the greedy algorithm 3oits flexibilities well and yields good solutions. More generally, our results suggest that already little time flexibilities can improve the overall system performance significantly.

I. Introduction

Today's datacenter networks are often largely oversubscribed (e.g., [1]), making network bandwidth a scarce resource shared across many tenants. The resulting contention of the different applications during their networking intensive phases, harms performance and renders running times unpredictable, potentially entailing a higher cost on the cloud customer side [2], [3].

The observation that high-performance applications also require high-performance networks connecting the distributed servers motivates a shift towards the network virtualization paradigm [4], [5]. A virtual network (VNet) supports the specification of bandwidth guarantees and provides performance isolation. These guarantees can be per VM-pair (e.g., the graph-based VNet topologies used in SecondNet [6]) or per VM (e.g., the hose-based networks of Oktopus [7]).

However, if specified too coarsely and statically for the entire execution, strict resource guarantees can come at the cost of wasting node and link resources, resulting in subpar utilization and a possible income loss for the datacenter operator. Indeed, many applications cycle through different phases, only some of which are network-intensive (e.g., the duce shuffle phase). The traffic patterns measured in [8] indicate that popular cloud applications only generate substantial traffic during only 30%-60% of the entire execution. Accordingly, VNets should support temporally varying specifications.

Besides the changing requirements over time, applications (and hence VNets) may also differ in their scheduling requirements: while some applications must be started immediately upon request, others may come with certain flexibilities on when they are executed, e.g. when combined with a corresponding price incentive [9].

In this paper we therefore initiate the study of the temporal VNet embedding problem (TVNEP) in which VNets must be embedded for a given duration and within a specified interval. Hence, the TVNEP conceptually consists of two tasks: (1) finding good embeddings for the VNets, and (2) scheduling the VNets in such a fashion, according to their temporal specification, that no resource capacities are exceeded. Solving the TVNEP is challenging, as already the efficient VNet embedding (i.e., mapping the virtual machines (VMs) of an application), is computationally hard.

Furthermore note that the TVNEP is not only relevant for data-centers and high performance computing applications, but also for wide-are networks (WANs). For example, Google recently presented its *B4 network* [10] that connects roughly a dozen datacenters using a Software-Defined Networking (SDN) approach: the bandwidth-intensive data copies from one site to another, are planned from the logically centralized perspective of the SDN controller. This allows to run the network at higher utilizations and to prioritize interactive applications during periods of failure or resource constraints. According to the authors, no more than a few dozen datacenter deployments are anticipated in the near future, which renders such a central control of bandwidth feasible.

a) Contribution: This paper initiates the study of the TVNEP problem. In contrast to existing literature focusing on VNet mappings only, the TVNEP asks for the *joint* optimization of *when* and *how* VNets shall be embedded. We present multiple mathematical programming formulations, and devise techniques to reduce the problems complexity to enable solving the TVNEP on moderately sized instances to optimality. Concretely, we make the following main contributions:

1) We show that interestingly, the TVNEP can be formu-

lated as a continuous-time model, which in contrast to discrete models facilitates a compact and accurate representation of time. We present two different Mixed Integer Program (MIP) formulations for this continuous model: the first, the so-called Δ-Model, is based on *state change* representations at events and the second, the Σ-Model, is based on explicit state representations. We discuss the (dis-)advantages of either of these approaches, and argue that Σ formulations give much better relaxations, hence speeding up the branch-and-bound algorithm employed to solve the model.

2) Our main technical contribution is the cΣ-Model: a very compact and improved Σ-Model which is based on rigorous state-space and symmetry reductions. The model clearly outperforms the other formulations, and enables us to solve moderately sized instances in the first place.
3) We show that the cΣ-Model cannot only be used to solve the TVNEP to optimality, but can also serve as a basis to develop fast heuristic algorithms. In particular, we describe the greedy heuristic $\text{c}\Sigma_{\text{A}}^{\text{G}}$, to perform the challenging objective of access control, i.e. maximizing the number of accepted VNets.
4) We report on our computational evaluation, and show that using the cΣ approach, solving the TVNEP to optimality is feasible for reasonable problem instances, and for different objectives, including access control, load balancing and makespan minimization. Furthermore, the cΣ-Model significantly outperforms the Δ- and the simple Σ-Model. Our evaluation also shows that $\text{c}\Sigma_{\text{A}}^{\text{G}}$ is often able to find good approximate solutions within seconds, making the algorithm an attractive alternative in situations where rigorous optimizations are not feasible.

b) Paper Organization: The remainder of this paper is organized as follows. We formally introduce the TVNEP problem in Section II. Section III discusses two main approaches to formulate continuous-time programs, and Section IV introduces the compact state formulation. The greedy algorithm is described in Section V. We report on our computational experiments in Section VI. After reviewing related work in Section VII, we conclude the work in Section IX.

c) Notation: If $E \subseteq V^2$ is a set of directed edges, then $\delta_E^+(v \in V) = (\{v\} \times V) \cap E$ and $\delta_E^-(v \in V) = (V \times \{v\}) \cap E$ denote the outgoing and the incoming edges respectively. When considering points in time, we use $(\cdot,\cdot)$ and $[\cdot,\cdot]$ to denote open and closed intervals respectively. We use $\mathbb{B} = \{0,1\}$ and use sans serif fonts for macros, i.e. textual substitutions.

II. The Temporal VNet Embedding Problem

The Temporal VNet Embedding Problem (TVNEP) extends the classic Virtual Network Embedding Problem (VNEP) (see [11] for a survey) by also considering *when* requests are to be scheduled. We first shortly revisit the VNEP.

A. The Virtual Network Embedding Problem

The input to the classic static VNEP is a set $\mathcal{R} = \{R_1, \dots, R_k\}$ of VNet requests that shall be embedded on a substrate network $\mathcal{S}$ with node and link resources (see Table I). The requests generally specify a VNet with resource requirements for both their nodes and links (see Table II) and without information on *where* to be mapped: this is the task of the embedding algorithm. Usually, requested node resources are memory, disk storage and CPU while links are described by means of bandwidth. In our problem formulation, we assume only a single node and a single link resource.

Table I
Definition and specification of the substrate

V_S	substrate nodes
$\text{E}_\text{S} \subseteq \text{V}_\text{S} \times \text{V}_\text{S}$	substrate links
$\text{c}_\text{S} : \text{V}_\text{S} \cup \text{E}_\text{S} \to \mathbb{R}^+$	substrate capacity

Table II
Static Request Parameters

$\forall \text{R} \in \mathcal{R}.$	V_R	virtual nodes of request R
$\forall \text{R} \in \mathcal{R}.$	$\text{E}_\text{R} \subseteq \text{V}_\text{R} \times \text{V}_\text{R}$	virtual links of request R
$\forall \text{R} \in \mathcal{R}.$	$\text{c}_\text{R} : \text{V}_\text{R} \cup \text{E}_\text{R} \to \mathbb{R}^+$	resources requestes by R

The output of an embedding algorithm is a mapping of the virtual nodes of a VNet to the substrate nodes, plus a mapping of the virtual links to corresponding physical path(s) (see Table III). Virtual links can either be embedded as a single unsplittable flow, or as a splittable multi-commodity flow. Intuitively, the closer two virtual nodes of a given VNet are mapped in the substrate network, the less resources are needed on the physical path connecting the two virtual nodes.

As our formulations for the TVNEP solve the VNEP as a subproblem, we will give now the constraints to find a local, i.e. time-independent, embedding for a request. Similarly to [12], we define the following variables (see Table III). As we consider splittable flows the mapping of virtual links onto substrate links is not binary, but real valued.

Table III
Variables for Embedding Requests

	$x_\mathcal{R} \colon \mathcal{R} \to \mathbb{B}$	decides which requests to embed
$\forall \text{R} \in \mathcal{R}.$	$x_V \colon \text{V}_\text{R} \times \text{V}_\text{S} \to \mathbb{B}$	maps virtual nodes on substrate nodes
$\forall \text{R} \in \mathcal{R}.$	$x_E \colon \text{E}_\text{R} \times \text{E}_\text{S} \to [0,1]$	maps virtual links on substrate links

To compute time-invariant embeddings, the following two constraints stated in Table IV suffice. By Constraint (1), all virtual nodes must be mapped onto a single substrate node, iff. the request is to be embedded. Constraint (2) constructs a splittable flow. Given a link $L_v = (N_v^+, N_v^-) \in \text{E}_\text{R}$ of request $\text{R} \in \mathcal{R}$, this constraint specifies flow preservation on all nodes on which neither N_v^+ nor N_v^- have been mapped. On the other hand, if N_v^+ has been mapped on N_s^+, then the flow balance is set to be -1 at node N_s; similarly if N_v^- has been mapped on N_s^- then the flow balance must equal 1, therefore constructing a unit flow from N_s^- to N_s^+.

For the VNEP to yield feasible solutions, it must be guaranteed that substrate capacities are not exceeded. To shorten notation, we use the macros presented in Table V for

computing the allocations of request $\mathrm{R} \in \mathcal{R}$ on substrate node $N_s \in \mathrm{V_S}$ or link $L_s \in \mathrm{E_S}$.

Table IV
MAPPING OF NODES AND LINKS ONTO THE SUBSTRATE

$$\forall \mathrm{R} \in \mathcal{R}. \forall N_v \in \mathrm{V_R}. \quad x_{\mathcal{R}}(\mathrm{R}) = \sum_{N_s \in \mathrm{V_S}} x_V(N_v, N_s) \tag{1}$$

$$\forall \mathrm{R} \in \mathcal{R}. \forall L_v = (N_v^+, N_v^-) \in \mathrm{E_R}. \forall N_s \in \mathrm{V_S} \quad \sum_{L_s \in \delta^+(N_s)} x_E(L_v, L_s) - \sum_{L_s \in \delta^-(N_s)} x_E(L_v, L_s) = x_V(N_v^-, N_s) - x_V(N_v^+, N_s) \tag{2}$$

Table V
MACROS

$$\forall \mathrm{R} \in \mathcal{R}. \forall N_s \in \mathrm{V_S}. \quad \mathsf{alloc_V}(R, N_s) = \sum_{N_v \in \mathrm{V_R}} \mathrm{c_R}(N_v) \cdot x_V(N_v, N_s)$$
$$\forall \mathrm{R} \in \mathcal{R}. \forall L_s \in \mathrm{E_S}. \quad \mathsf{alloc_E}(R, L_s) = \sum_{L_v \in \mathrm{E_R}} \mathrm{c_R}(L_v) \cdot x_E(L_v, L_s)$$

Several objectives have been proposed for the VNEP [11]: a natural criterion is to try to embed as many VNets as possible; i.e., the embedding algorithm additionally performs *access control* by deciding which VNets to accept and which to reject. Another criterion is to embed a given set of VNets (thus fixing $x_{\mathcal{R}}(\mathrm{R}) = 1$ for all requests $\mathrm{R} \in \mathcal{R}$) in a way that minimizes the maximal link load, or that uses the least physical nodes or links (e.g., for energy saving considerations).

B. The Temporal Virtual Network Embedding Problem

The TVNEP regards the question of *where* and *when* to embed a VNet: besides the VNet's resource specification (see Table II), VNets are attributed with three temporal parameters: the duration of execution d, the earliest possible start and latest end point in time, t^s and t^e respectively (see Table VI).

Table VI
TEMPORAL REQUEST PARAMETERS

	$\mathrm{T} > 0$	maximal considered time horizon
$\forall \mathrm{R} \in \mathcal{R}.$	$\mathrm{t_R^s} \geq 0$	earliest possible start of R
$\forall \mathrm{R} \in \mathcal{R}.$	$\mathrm{t_R^e} \leq T$	latest possible end of R
$\forall \mathrm{R} \in \mathcal{R}.$	$\mathrm{d_R} \in \mathbb{R}^+$	duration of request R

Clearly, if $t^e - t^s > d$ the provider may harness this temporal flexibility by scheduling the VNet's execution such that e.g. bottlnecks are averted. While the provider may schedule the VNet arbitrarily subject to lying in the interval $[t^s, t^e]$, we assume that the embedding of the VNet onto the substrate is invariant, i.e. does not change, over time. Even though a model in which the provider may arbitrarily migrate VNets might be of interest in its own right, such reconfigurations are likely to incur additional resource allocations, e.g. to transfer the data of one VM to another substrate node. Note however, that our model presented in Section IV can be easily adapted to model *explicit* migrations [13].

Definition 2.1 (Temporal VNet Embedding Problem):
Given: A capacitated substrate network $\mathcal{S}$, a set of requests $\mathcal{R}$ and a time horizon $\mathrm{T} > 0$ (see Tables I,II,VI).
Task: Find an optimal temporal embedding, consisting of a static embedding $x_{\mathcal{R}}, x_V, x_E$ (see Table III), and points in time for each request $\mathrm{R} \in \mathcal{R}$ to start and end $t_R^+, t_R^- \in [0, T]$, such that:

1) the static embedding $x_{\mathcal{R}}, x_V, x_E$ satisfies the Constraints (1) and (2) (see Table IV),
2) $t_R^- - t_R^+ = d_R$, $t_R^s \leq t_R^+$ and $t_R^- \leq t_R^e$ holds and
3) for all points in time $t \in [0, T]$ allocations are feasible:

$$\forall N_s \in \mathrm{V_S}. \ \mathrm{c_S}(N_s) \geq \sum_{\substack{\mathrm{R} \in \mathcal{R} \text{ with} \\ t \in (t_\mathrm{R}^+, t_\mathrm{R}^-)}} \mathsf{alloc_V}(\mathrm{R}, N_s) \ ,$$
$$\forall L_s \in \mathrm{E_S}. \ \mathrm{c_S}(L_s) \geq \sum_{\substack{\mathrm{R} \in \mathcal{R} \text{ with} \\ t \in (t_\mathrm{R}^+, t_\mathrm{R}^-)}} \mathsf{alloc_E}(\mathrm{R}, L_s) \ .$$

In the above definition, we did not specify a concrete objective function to optimize the temporal embedding for, as several different ones will be considered. The objective functions proposed in this paper comprise access control, load balancing on nodes, disabling of links to reduce energy consumption and maximizing the earliness of requests. As further notation related with our cΣ-Model will be necessary to state these, we will present the objectives in Section IV-E.

III. THE CONTINUOUS-TIME APPROACH

This section shows that the TVNEP problem can be modeled using a continuous-time approach. This is attractive as it avoids inaccuracies due to time discretizations and therefore allows us to solve the continuous VNEP as stated in Definition 2.1. We begin by discussing the conceptual model of abstract event points, and then derive two complementary ways to model the TVNEP in our continuous-time framework: the Δ-Model and the Σ-Model. While the Δ-Model only represents state changes and therefore requires less variables, the Σ-Model introduces explicit state variables to improve the LP-relaxations. We will argue that the latter is preferable, and it will also build the basis of our optimized cΣ-Model presented in the subsequent section.

Our mathematical programming formulations rely on the event point model introduced in the following, which will allow to check the feasibility of possible solutions to the TVNEP.

A. The Abstract Event Point Model

To check whether resource allocations hold at each point in time $t \in [0, T]$, it suffices to consider the $2 \cdot |\mathcal{R}| - 1$ many intervals in which resource allocations are invariant: letting $\mathcal{T} = \{t_R^+, t_R^- | R \in \mathcal{R}\}$, these intervals are defined by $\{[t^-, t^+] | t^-, t^+ \in \mathcal{T}, (t^-, t^+) \cap \mathcal{T} = \emptyset\}$. Our models compute these time-invariant states using an event point model illustrated in Figure 1. We define the set of events $\mathcal{E} = \{\mathrm{e}_1, \ldots, \mathrm{e}_{2\cdot|\mathcal{R}|}\}$ and the set of states $\mathcal{S} = \{\mathrm{s}_1, \ldots, \mathrm{s}_{2\cdot|\mathcal{R}|-1}\}$ and introduce the variables χ_R^+ and χ_R^- (see Table VII).

In our basic models, we require the mapping of the start and end of requests onto event points to be bijective. This can easily be modeled by the two following constraints:

$$\forall \mathrm{R} \in \mathcal{R}. \sum_{\mathrm{e}_i \in \mathcal{E}} \chi_\mathrm{R}^+(\mathrm{e}_i) = 1 \wedge \sum_{\mathrm{e}_i \in \mathcal{E}} \chi_\mathrm{R}^-(\mathrm{e}_i) = 1$$
$$\forall \mathrm{e}_i \in \mathcal{E}. \sum_{\mathrm{R} \in \mathcal{R}} \chi_\mathrm{R}^+(\mathrm{e}_i) = 1 \wedge \sum_{\mathrm{R} \in \mathcal{R}} \chi_\mathrm{R}^-(\mathrm{e}_i) = 1$$

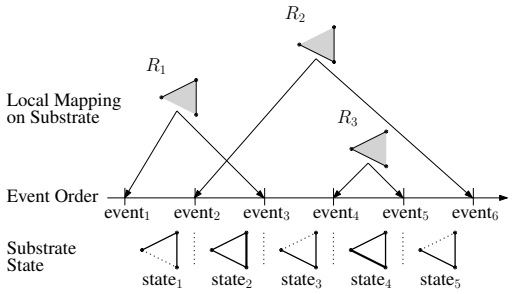

Figure 1. Shown are local substrate allocation of requests R_1, R_2, R_3 on a three node substrate. By assigning the start and end of requests to event points, states between events can be reconstructed to check feasibility of allocations.

Table VII
DEFINITION OF ABSTRACT EVENT POINT AND STATE SETS

$\forall \mathrm{R} \in \mathcal{R}.\ \chi^{+}_{\mathrm{R}} : \mathcal{E} \to \mathbb{B}$	maps the start of R on an event
$\forall \mathrm{R} \in \mathcal{R}.\ \chi^{-}_{\mathrm{R}} : \mathcal{E} \to \mathbb{B}$	maps the end of R on an event

By the order on the event points, the above mapping in turn defines a strict linear order on the start and end of the requests, therefore allowing to reconstruct the states between these event points. As we will discuss in the course of presenting the cΣ-Model (see Section IV), these event points will be attributed (ordered) time values and, via the mapping onto them, will specify the start and end time of the requests.

In the following we will now present two complementary approaches to reconstruct the states.

B. Δ-Model: Representing only State Changes

A first and intuitive way to formulate the continuous-time program, is to represent state allocations (at states $\mathcal{S}$), by encoding only the state *differences* or *changes* $\Delta_{\mathrm{e}_i} : \mathrm{V_S} \cup \mathrm{E_S} \to \mathbb{R}$ at event point e_i. We will refer to this model as the Δ-Model. Given these changes, the state allocations at state s_i computes to $\sum_{j=1}^{i} \Delta_{\mathrm{e}_i}$. However, to compute the Δ_{e_i} variables, conditional assignments of the following form are necessitated (exemplified for substrate node $N_s \in \mathrm{V_S}$)

$$\Delta_{\mathrm{e}_i}(N_s) = \begin{cases} +\mathsf{alloc_V}(R, N_s) & \text{, if } \chi^{+}_{R_1}(\mathrm{e}_i) = 1 \\ -\mathsf{alloc_V}(R, N_s) & \text{, if } \chi^{-}_{R_1}(\mathrm{e}_i) = 1 \\ \vdots & \\ +\mathsf{alloc_V}(R, N_s) & \text{, if } \chi^{+}_{R_k}(\mathrm{e}_i) = 1 \\ -\mathsf{alloc_V}(R, N_s) & \text{, if } \chi^{-}_{R_k}(\mathrm{e}_i) = 1 \end{cases}$$

This type of selection constraint can, in the framework of Mixed-Integer Programming, only be modeled in the following way:

$$\Delta_{\mathrm{e}_i}(N_s) \leq +\mathsf{alloc_V}(R, N_s) + \mathrm{c_S}(N_s)(1 - \chi^{+}_{R_1}(\mathrm{e}_i)) \quad (3)$$

$$\Delta_{\mathrm{e}_i}(N_s) \geq +\mathsf{alloc_V}(R, N_s) - \mathrm{c_S}(N_s)(1 - \chi^{+}_{R_1}(\mathrm{e}_i)) \cdot 2 \quad (4)$$

$$\Delta_{\mathrm{e}_i}(N_s) \leq -\mathsf{alloc_V}(R, N_s) + \mathrm{c_S}(N_s)(1 - \chi^{-}_{R_1}(\mathrm{e}_i)) \cdot 2 \quad (5)$$

$$\Delta_{\mathrm{e}_i}(N_s) \geq -\mathsf{alloc_V}(R, N_s) - \mathrm{c_S}(N_s)(1 - \chi^{-}_{R_1}(\mathrm{e}_i)) \quad (6)$$

As shown in the computational evaluation, the utilization of these constraint types yields very weak models, even though the number of variables is considerably smaller. We will now give an example, that motivates the explicit state representation of the Σ-Model.

In the branch-and-bound process used to solve MIP formulations, non-fixed binary variables can attain any value in the range $[0, 1]$. Assume now that, besides other requests, two long lasting requests R_1, R_2 consisting of single nodes ($\mathrm{V}_{R_1} = \{v_1\}, \mathrm{V}_{R_2} = \{v_2\}$), are to be embedded on a single node substrate ($\mathrm{V_S} = \{s\}$) and that both require all resources on this node ($\mathrm{c}_{R_1}(v_1) = \mathrm{c}_{R_2}(v_2) = \mathrm{c_S}(s)$). Clearly, such an embedding is not possible if R_1 and R_2 overlap temporally. The relaxation of the event mapping variables however allows for the following setting: $\chi^{+}_{R_1}(\mathrm{e}_j) = \chi^{+}_{R_2}(\mathrm{e}_j) = 0.5$. Under this assignment, the above Constraints (3) and (4) reduce to $0 \leq \Delta_{\mathrm{e}_j}(s) \leq \mathrm{c_S}(s)$ for $j \in \{1, 2\}$. Thus, in the relaxation the variables $\Delta_{\mathrm{e}_j}(s)$ can attain the value 0 and therefore no state allocations for R_i will ever become visible in the substrate's state. Furthermore, as the variables Δ_{e_j} allow also for negative state changes, under the mapping $\chi^{-}_{R_1}(\mathrm{e}_l) = \chi^{-}_{R_2}(\mathrm{e}_l) = 0.5$ for $l \in \{101, 102\}$ the constraints (5) and (6) reduce to $-\mathrm{c_S}(s) \leq \Delta_{\mathrm{e}_j}(s) \leq 0$ and therefore all previous resource allocations accounted for at state s_{100} on node s can be effectively nullified.

C. Σ-Model: Representing States Explicitly

As discussed above, the linear relaxation of the Δ-Model may fail to account for any resource allocations previously made. To ensure better relaxations, we will now present the Σ-Model that explicitly represents states at the cost of introducing $\mathcal{O}(|\mathcal{S}| \cdot |\mathcal{R}|)$ additional variables to represent local state allocations made by each request. In Section IV we will then show how the state-space can be effectively reduced.

To compute the local allocations, we use macro $\Sigma(R, \mathrm{e}_i)$ defined in Table VIII. Intuitively $\Sigma(R, \mathrm{e}_i)$, computes for any event point $\mathrm{e}_i \in \mathcal{E}$ to which extent request $R \in \mathcal{R}$ is embedded. This allows us to compute resource allocations as defined in Table IX: if request $\mathrm{R} \in \mathcal{R}$ is not embedded at event e_i, i.e. $\Sigma(R, \mathrm{e}_i) = 0$, then the local state allocations a_{R} are not constrained, while if $\Sigma(R, \mathrm{e}_i) = 1$ holds, the local state allocations are lower bounded by the actual resource usage. Finally, the Constraint (9) of Table IX guarantees feasibility of all state allocations by upper bounding the sum of all local state allocations for the resources $r \in \mathrm{V_S} \cup \mathrm{E_S}$ by its respective capacity $\mathrm{c_S}(r)$.

Revisiting the single substrate example from the above section, we note that by the above model $\Sigma(R_i, \mathrm{e}_k) = 1$ will hold for $i = \{1, 2\}$ and $k = \{3, \ldots, 100\}$, implying that $a_{R_i}(\mathrm{s}_k, s) \geq \mathrm{c_S}(s)$ will hold, showing that $\chi^{+}_{R_i}(\mathrm{e}_i) = 0.5$ and $\chi^{+}_{R_i}(\mathrm{e}_{i+100}) = 0.5$ is an infeasible relaxation. The Σ-Model is therefore provably stronger than the Δ-Model, as it excludes linear relaxation that are feasible for the Δ-Model. Since the superiority of this model is based on the ability to derive state allocation directly using $\Sigma(R, \mathrm{e}_i)$, we refer to this model as Σ-Model.

Table VIII
STATE ALLOCATION VARIABLES FOR REQUESTS

$a_{\mathrm{R}} : \mathcal{S} \times (\mathrm{V_S} \cup \mathrm{E_S}) \to \mathbb{R}_{\geq 0}$ allocations of request R over states

$\forall \mathrm{R} \in \mathcal{R}. \forall \mathrm{e}_i \in \mathcal{E}.\ \ \boldsymbol{\Sigma}(R, \mathrm{e}_i) = \sum_{j=1,\ldots,i} \chi^{+}_{\mathrm{R}}(\mathrm{e}_j, \mathrm{R}) - \sum_{j=i,\ldots,|\mathcal{E}|} \chi^{-}_{\mathrm{R}}(\mathrm{e}_j, \mathrm{R})$

Table IX
COMPUTING LOCAL STATE ALLOCATIONS AND GUARANTEEING FEASIBILITY

$$\forall \mathrm{R} \in \mathcal{R}. \forall \mathrm{s}_i \in \mathcal{S}. \forall L_s \in \mathrm{E_S}. \quad a_{\mathrm{R}}(\mathrm{s}_i, L_s) \geq \mathsf{alloc}_{\mathsf{E}}(\mathrm{R}, L_s) - \mathrm{c_S}(L_s) \cdot (1 - \boldsymbol{\Sigma}(R, \mathrm{e}_i)) \tag{7}$$

$$\forall \mathrm{R} \in \mathcal{R}. \forall \mathrm{s}_i \in \mathcal{S}. \forall N_s \in \mathrm{V_S}. \quad a_{\mathrm{s}_i}(\mathrm{R}, N_s) \geq \mathsf{alloc}_{\mathsf{V}}(\mathrm{R}, N_s) - \mathrm{c_S}(N_s) \cdot (1 - \boldsymbol{\Sigma}(R, \mathrm{e}_i)) \tag{8}$$

$$\forall \mathrm{s}_i \in \mathcal{S}. \forall r \in \mathrm{V_S} \cup \mathrm{E_S}. \qquad \mathrm{c_S}(r) \geq \sum_{\mathrm{R} \in \mathcal{R}} a_{\mathrm{R}}(\mathrm{s}_i, r) \tag{9}$$

IV. THE COMPACT STATE MODEL cΣ

Based on the discussions and insights from the Δ-Model and Σ-Model discussed in the previous section, we have developed the optimized and compact cΣ-Model.

We begin by outlining the major improvement over the Σ-Model, namely the reduction of symmetries and the state-space by only employing $|\mathcal{R}|+1$ many event points compared to the $2 \cdot |\mathcal{R}|$ many event points used in the Δ- and Σ-Model. As the embedding of requests and the general check of feasibility was already introduced in Sections II and III we present how to assign points in time to event points and how these relate to the requests' start and end time. After having presented the cΣ-Model in full, we state the objective functions used in our computational evaluation and then turn towards another very important technique to reduce the state-space and improve the LP-relaxation, the *temporal dependency graph cuts*.

A. Compactification

The major enhancement of the cΣ- over the Σ-Model lies in the following observation: to check whether a state is feasible, it suffices to consider *only* the states originating from the start of a request. The truth of this is immediate, as the end of a request may only reduce state allocations.

Figure 2 depicts the same scenario as Figure 1 but in the compactified model. Here, each start of a request must be assigned uniquely to one event point. As the ordering of the ends of requests does not change the feasibility, we allow for the assignment of multiple requests' ends to the same event point. The semantic of our model will be the following: if the end of a request is mapped onto an event point e_i, then it must have ended between event points e_{i-1} and e_i.

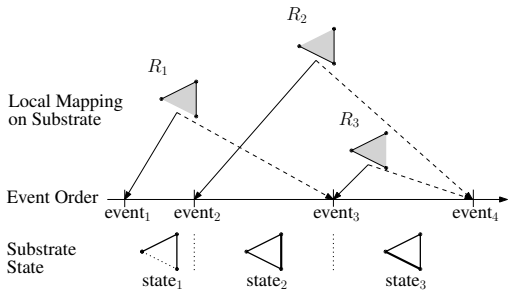

Figure 2. The example of Figure 1 revisited. By using only $|\mathcal{R}|$+1 many event points and allowing to map multiple requests' ends on events, the state model can be compactified.

Table X
DEFINITION OF ABSTRACT EVENT POINT AND STATE SETS

$\mathcal{E} = \{\mathrm{e}_1, \mathrm{e}_2 ..., \mathrm{e}_{|\mathcal{R}|+1}\}$ abstract event points
$\mathcal{S} = \{\mathrm{s}_1, \mathrm{s}_2, ..., \mathrm{s}_{|\mathcal{R}|}\}$ states between event points

Table XI
ASSOCIATING REQUEST GROUPS WITH EVENTS

$$\forall \mathrm{R} \in \mathrm{RG}. \quad \sum_{\mathrm{e}_i \in \{\mathrm{e}_1, \ldots, \mathrm{e}_{|\mathcal{R}|}\}} \chi^{+}_{\mathrm{R}}(\mathrm{e}_i) = 1 \tag{10}$$

$$\sum_{\mathrm{e}_i \in \{\mathrm{e}_2, \ldots, \mathrm{e}_{|\mathcal{R}|+1}\}} \chi^{-}_{\mathrm{R}}(\mathrm{e}_i) = 1 \tag{11}$$

$$\forall \mathrm{e}_i \in \{\mathrm{e}_1, \ldots, \mathrm{e}_{|\mathcal{R}|}\}. \quad \sum_{\mathrm{R} \in \mathcal{R}} \left(\chi^{+}_{\mathrm{R}}(\mathrm{e}_i)\right) = 1 \tag{12}$$

The set of events and states is defined in Table X while the constraints assigning requests to event points are given in Table XI.

The advantage of the cΣ-Model over the Σ-Model is twofold. Firstly, halving the number of states, halves the number of variables needed for requests' local state allocations (*state-space reduction*). Secondly, the cΣ-Model can effectively reduce *symmetries* as discussed in Section IV-D.

B. Incorporating Time into the cΣ-Model

We will now present how time can be incorporated into the cΣ-Model. As the approach presented here generally applies to the Δ- and Σ-Models too, we only present the cΣ-version constraints. Table XII introduces the variables used.

Table XIII states the constraints used to enforce temporal feasibility. First we impose a (weak) monotonic order on the events' points time values. The Constraints (14) and (15) set the start time of request $\mathrm{R} \in \mathcal{R}$ to the event point it is associated with. Note that using the sum is valid here, as we sum only over earlier and later events respectively. The Constraints (16) and (17) are a slight adaption of (14) and (15), as in the lower bound for t^{-}_{R} the time point of the previous event $t_{\mathrm{e}_{i-1}}$ is used. This is necessary, as when $\chi^{-}_{\mathrm{R}}(\mathrm{e}_i) = 1$ holds for some $\mathrm{R} \in \mathcal{R}$ and $\mathrm{e}_i \in \{\mathrm{e}_2, \ldots, \mathrm{e}_{|\mathcal{R}|+1}\}$, then R will end in the interval $[t_{\mathrm{e}_{i-1}}, t_{\mathrm{e}_i}]$. Lastly, the last constraint of Table XIII requires that all requests are embedded for the requested duration.

Together with the parameters, variables and constraints found in Sections II and III the Tables I-XIII present a complete formulation for the TVNEP. However, in the next section, we will introduce further constraints to strengthen the model, which will complete the description of the cΣ-Model.

Table XII
TEMPORAL VARIABLES

$\forall \mathrm{e}_i \in \mathcal{E}.\ \ t_{\mathrm{e}_i} \in \mathbb{R}_{\geq 0}$ time point at which event e_i takes place
$\forall \mathrm{R} \in \mathcal{R}.\ \ t^{+}_{\mathrm{R}} \in \mathbb{R}_{\geq 0}$ point in time at which the request R is embedded
$\forall \mathrm{R} \in \mathcal{R}.\ \ t^{-}_{\mathrm{R}} \in \mathbb{R}_{\geq 0}$ point in time at which R's embedding ends

Table XIII
ASSOCIATING REQUEST GROUPS WITH EVENTS

$$\forall \mathrm{e}_i \in \{\mathrm{e}_1, \dots, \mathrm{e}_{|\mathcal{R}|}\}. \qquad t_{\mathrm{e}_i} \leq t_{\mathrm{e}_{i+1}} \tag{13}$$

$$\forall \mathrm{R} \in \mathcal{R}. \forall \mathrm{e}_i \in \{\mathrm{e}_1, \dots, \mathrm{e}_{|\mathcal{R}|}\}.$$
$$t^+_{\mathrm{R}} \leq t_{\mathrm{e}_i} + (1 - \sum_{j=1,\dots,i} \chi^+_{\mathrm{R}}(\mathrm{e}_j, \mathrm{R})) \cdot \mathrm{T} \tag{14}$$
$$t^+_{\mathrm{R}} \geq t_{\mathrm{e}_i} - (1 - \sum_{j=i,\dots,|\mathcal{E}|} \chi^+_{\mathrm{R}}(\mathrm{e}_j, \mathrm{R})) \cdot \mathrm{T} \tag{15}$$

$$\forall \mathrm{R} \in \mathcal{R}. \forall \mathrm{e}_i \in \{\mathrm{e}_2, \dots, \mathrm{e}_{|\mathcal{R}|+1}\}.$$
$$t^-_{\mathrm{R}} \leq t_{\mathrm{e}_i} + (1 - \sum_{j=2,\dots,i} \chi^-_{\mathrm{R}}(\mathrm{e}_j, \mathrm{R})) \cdot \mathrm{T} \tag{16}$$
$$t^-_{\mathrm{R}} \geq t_{\mathrm{e}_{i-1}} - (1 - \sum_{j=i,\dots,|\mathcal{E}|} \chi^-_{\mathrm{R}}(\mathrm{e}_j, \mathrm{R})) \cdot \mathrm{T} \tag{17}$$

$$\forall \mathrm{R} \in \mathcal{R}. \qquad \mathrm{d}_{\mathrm{R}} = t^-_{\mathrm{R}} - t^+_{\mathrm{R}} \tag{18}$$

C. Temporal Dependency Graph Cuts

Considering the cΣ-Model presented above, one finds that both temporal as well as state allocation variables are constrained using the event mapping variables $\chi^+_{\mathrm{R}}, \chi^-_{\mathrm{R}}$. To yield reasonable LP-relaxations, the event mapping variables should therefore be as less *smeared* as possible, i.e. that requests' starts and ends should be assigned fractionally only to as few event points as possible. In this section, we therefore introduce *Temporal Dependency Graph Cuts*: these cuts are valid constraints that can be a priori derived from the requests' temporal specification and strengthen the model.

We introduce the directed temporal dependency graph $G_{\mathsf{dep}}(\mathcal{R}) = (V_{\mathsf{dep}}, E_{\mathsf{dep}})$ that will reflect *temporal* dependencies. We define the graph as follows. The set of nodes represents the *abstract* start and end point for each request: $V_{\mathsf{dep}} = \mathcal{R} \times \{\mathsf{start}, \mathsf{end}\}$. We define the following functions to calculate the earliest possible start and the latest possible end time.

$$\mathsf{earliest}((\mathrm{R}, \mathsf{t}) \in V_{\mathsf{dep}}) = \begin{cases} t^s_R & , \text{if } \mathsf{t} = \mathsf{start} \\ t^s_R + \mathrm{d}_{\mathrm{R}} & , \text{if } \mathsf{t} = \mathsf{end} \end{cases}$$

$$\mathsf{latest}((\mathrm{R}, \mathsf{t}) \in V_{\mathsf{dep}}) = \begin{cases} t^e_R - \mathrm{d}_{\mathrm{R}} & , \text{if } \mathsf{t} = \mathsf{start} \\ t^e_R & , \text{if } \mathsf{t} = \mathsf{end} \end{cases}$$

A directed edge $(v, w) \in V^2_{\mathsf{dep}}$ will be contained in E_{dep} iff. v *must* start *before* w:

$$E_{\mathsf{dep}} = \{(v, w) \in V^2_{\mathsf{dep}} | \mathsf{latest}(v) < \mathsf{earliest}(w)\}.$$

Having defined the dependency graph, we can now start to derive cuts, i.e. valid inequalities. First convince yourself that $G_{\mathsf{dep}}(\mathcal{R})$ is acyclic. For $(v, w) \in E_{\mathsf{dep}}$, we define the weight of (v, w) to be 1, if v represents the start of a request and 0 otherwise. As $G_{\mathsf{dep}}(\mathcal{R})$ is acyclic, we can compute maximal distances by negating the weights and applying the Floyd-Warshall algorithm [14]. Let $\mathsf{dist}_{\max} : V_{\mathsf{dep}} \times V_{\mathsf{dep}} \to \mathbb{N}$ denote the maximal distances between any two nodes. We set $\mathsf{dist}_{\max}(v, w) = 0$ if w is not reachable from v in $G_{\mathsf{dep}}(\mathcal{R})$. We make the following observations:

1) If a node $v \in V_{\mathsf{dep}}$ *is reachable* from n many nodes $\{(\mathrm{R_i}, \mathsf{start})\}$, $1 \leq i \leq n$ then v cannot be mapped on either one of the first n events.
2) Similarly: If a node $v \in V_{\mathsf{dep}}$ *reaches* n other nodes $\{(\mathrm{R_i}, \mathsf{start})\}$, $1 \leq i \leq n$ then all these must occur *after* v. Furthermore, if v itself is a start event, then its corresponding end event must be mapped after the start. Therefore, if $v = (\mathrm{R}, \mathsf{start})$, then v cannot be mapped on the last $n + 1$ events and otherwise, if $v = (\mathrm{R}, \mathsf{end})$ then v cannot be mapped on the last n events.
3) Let $v, w \in V_{\mathsf{dep}}$, such that $0 < \mathsf{dist}_{\max}(v, w) = d$ holds. Assuming that w is mapped on e_i then v must be mapped on $\{\mathrm{e}_1, ..., \mathrm{e}_{i-d}\}$.

Note that the first two above observations can be formulated in the following graph theoretical way: given a node $v \in V_{\mathsf{dep}}$ consider the maximal subgraph (with respect to nodes and edges) of G_{dep} such that all nodes within this subgraph reach v or can be reached by v respectively. Counting the number of edges attributed with 1 then directly gives the number of leading and trailing event points on which the request of v cannot be embedded. Denoting these values by $\mathsf{dist}^+_{\max}(v)$ and $\mathsf{dist}^-_{\max}(v)$ we can derive the Temporal Dependency Graph Cuts (see Table XIV), in which we use the following macro:

$$\chi_{\mathsf{Event}}(\mathrm{e}_i \in \mathcal{E}, (\mathrm{R}, \mathsf{t}) \in V_{\mathsf{dep}}) = \begin{cases} \chi^+_{\mathrm{R}}(\mathrm{e}_i) & \text{if } \mathsf{t} = \mathsf{start} \\ \chi^-_{\mathrm{R}}(\mathrm{e}_i) & \text{if } \mathsf{t} = \mathsf{end} \end{cases}$$

Note that the above considerations hold also for the general Σ-Model, such that only slight modifications are necessary to construct the temporal dependency graph cuts for these models. While Constraint (20) can reduce LP smearings, Constraint (19) is of upmost importance to presolve the Σ- and cΣ-Model: as the event mapping variables are restricted to certain *event ranges* we can apply the following presolving routine if the start and end event ranges do not overlap. Let $\mathcal{E}^+_{\mathrm{R}} = \{e_i, ..., e_j\}$ and $\mathcal{E}^-_{\mathrm{R}} = \{e_{j+k}, ..., e_l\}$ denote the event ranges for the start and end of R respectively. Clearly, $\Sigma(R_{\mathrm{R}}, \mathrm{e}_n) = 1$ (see Table VIII) holds for $i \leq n < j + k$ and therefore we do not need to compute state allocations a_{R} for the intermediate states s_n but can directly factor the allocations of R into the sum of Constraint (9) (see Table IX). Constraint (19) therefore poses another important state-space reduction.

Table XIV
TEMPORAL DEPENDENCY GRAPH CUTS

$$\forall v \in V_{\mathsf{dep}}. \qquad \sum_{i=|\mathsf{dist}^+_{\max}(v)|+1}^{|\mathcal{R}|+1-|\mathsf{dist}^-_{\max}(v)|} \chi_{\mathsf{Event}}(\mathrm{e}_i, v) = 1 \tag{19}$$

$$\forall v \in V_{\mathsf{dep}}. \forall w \in \mathsf{dist}^+_{\max}(v). \forall \mathrm{e}_i \in \mathcal{E}, \mathsf{dist}_{\max}(v, w) + 1 \leq i \leq |\mathcal{R}|.$$
$$\textstyle\sum_{j=1}^{i} \chi_{\mathsf{Event}}(\mathrm{e}_j, w) \leq \sum_{\substack{\mathrm{e}_j \in \mathcal{E} \\ j \leq i - \mathsf{dist}_{\max}(v,w)}} \chi_{\mathsf{Event}}(\mathrm{e}_j, v) \tag{20}$$

D. Symmetry Reductions

We will now show that the compactification of the Σ-Model into the cΣ-Model yields symmetry reduction. Consider the scenario of k requests $\mathcal{R} = \{R_1, \dots, R_k\}$ of duration $\mathrm{d}_k = 1 + 1/2^k$ which are to be embedded in the interval $[0, 2]$. Clearly, a priori, the only possible event order is to first start

all requests and then end them (this is indeed implied by the temporal dependency graph cuts of Table XIV). We will now show that the number of possible solutions can be reduced by a factor of 2^k using the cΣ-Model in comparison to the Δ- and Σ-Model. Assume that the requests' starts are assigned in the order of ascending durations: $\chi^+_{R_i}(\mathrm{e}_i) = 1$. Based on the chosen durations, R_{i+1} can either be scheduled to complete before R_i or after R_i by setting $t^-_{R_{i+1}} = t^-_{R_i} \pm 1/2^{k+1}$. Therefore, having fixed the requests' start variables, there might be as much as 2^k many possible combinations to order the requests' ends. In contrast, the cΣ-Model allows only for a single solution, in which all requests' ends are mapped on the last event point.

E. Objective Functions

We will now present the objective functions used in the evaluation of our approach.

1) Access Control: The task is to maximize the revenue of the substrate's provider. We choose the revenue to be the sum requested virtual node resources over time:

$$\max \sum_{\mathrm{R} \in \mathcal{R}} x_{\mathcal{R}}(\mathrm{R}) \cdot \mathrm{d_R} \cdot \sum_{N_v \in \mathrm{V_R}} \mathrm{c_R}(N_v)$$

2) Maximize Earliness: Given a fixed set of requests to be embedded, the provider may charge an additional fee depending on how soon the request is embdded. We assume that this fee is proportional to the duration of the request, such that if the request is started at the earliest possible time, the fee is $\mathrm{d_R}$ and if the request starts at the latest possible time, the fee is 0.

$$\max \sum_{\mathrm{R} \in \mathcal{R}} \mathrm{d_R} \cdot (1 - \frac{t^+_\mathrm{R} - t^s_R}{t^e_R - \mathrm{d_R} - t^s_R})$$

3) Balance Node Load over Time: Given a fixed set of requests to be embedded, the provider may try to schedule and embed the given requests in such a way, that during the span of time under consideration, the number of nodes never being used more than some fraction of their capacity is maximized. To this end, we introduce additional binary variables $F : \mathrm{V_S} \to \mathbb{B}$, deciding whether a substrate node is never used more than $f\%$ of its capacity.

$$\max \sum_{N_s \in \mathrm{V_S}} F(N_s) \text{ , s.t. } \forall N_s \in \mathrm{V_S}. \forall \mathrm{s}_i \in \mathcal{S}.$$
$$(1 - F(N_s)) \cdot (1 - f) \cdot \mathrm{c_S}(N_s) \geq \sum_{\mathrm{R} \in \mathcal{R}} a_{\mathrm{s}_i}(N_s) - f \cdot \mathrm{c_S}(N_s)$$

4) Disable Links for Energy Savings: Another goal one may pursue with is the conservation of energy. Concretely, one may want to turn off as many links (i.e., switch or router ports) as possible for as long as possible [15]. To model this objective function, we introduce decision variables $D : \mathrm{E_S} \to \mathbb{B}$, such that $D(L_s) = 1$ iff. L_s can be disabled over the whole time span $[0, T]$.

$$\max \sum_{L_s \in \mathrm{E_S}} D(L_s) \text{ , } s.t.$$
$$\forall L_s \in \mathrm{E_S}. \sum_{\mathrm{R} \in \mathcal{R}. L_v \in \mathrm{E_R}} x_E(L_v, L_s) \leq |\mathcal{R}| * (1 - D(L_s))$$

V. Greedy Algorithm c$\Sigma^{\mathrm{G}}_{\mathrm{A}}$

The objective of maximizing the number of embeddable VNets (i.e., the access control objective) is a natural and important one, as it e.g., maximizes the revenue for the infrastructure provider. However, compared to other objectives with a fixed set of requests, it is considerably more complex. To complement our study of optimal algorithms, in this section, we will present a fast (and polynomial-time) greedy algorithm to compute heuristic solutions for this particular objective. Interestingly, the algorithm is still based on our cΣ-Model.

Algorithm c$\Sigma^{\mathrm{G}}_{\mathrm{A}}$ takes as input a TVNEP instance with fixed node mappings; alternative embeddings could be computed e.g. by employing the approach presented in [12]. However, we do not follow this direction further here and focus on the link embedding as well as the temporal scheduling aspects. First the set of requests is ordered according to the earliest point in time at which they might start. According to this order, Algorithm c$\Sigma^{\mathrm{G}}_{\mathrm{A}}$ iteratively tries to embed the request at hand using the cΣ-Model. The objective guarantees that the request is embedded, if possible; furthermore, if the request can be embedded, then it is started as early as possible. If a request has been accepted, it has to be embedded at every following iteration (by placing it in $\mathcal{R}^+$) and at exactly the points in time returned by the optimal solution computed. If a request is not embedded, its start and end times are fixed nevertheless (as required by Definition 2.1). Importantly, the algorithm *does not*

Algorithm c$\boldsymbol{\Sigma^{\mathrm{G}}_{\mathrm{A}}}$

Input : Substrate $\mathcal{S}$, Requests $\mathcal{R}$ (see Tables I, II,VI), node mappings $\forall \mathrm{R} \in \mathcal{R}.\ x'_V : \mathrm{V_R} \times \mathrm{V_S} \to \mathbb{B}$
Output: Solution to the TVNEP: $(\hat{x}_\mathcal{R}, \hat{x}_V, \hat{x}_E, \hat{t}^+, \hat{t}^-)$
begin
  set $\mathcal{R}', \mathcal{R}^+, \mathcal{R}^- \triangleq \emptyset$
  set $L \leftarrow \mathcal{R}$ ordered according to t^s_R
  for $i = 1$ **to** $|L|$ **do**
    $\mathcal{R}' \leftarrow \mathcal{R}' \cup \{L[i]\}$
    $(\hat{x}_\mathcal{R}, \hat{x}_V, \hat{x}_E, \hat{t}^+, \hat{t}^-) \leftarrow$ optimal solution of:
      max $\mathrm{T} \cdot x_\mathcal{R}(L[i]) + (\mathrm{T} - t^-_{L[i]})$ (21)
      s.t. c$\Sigma(\mathcal{R}')$ (22)
        $\forall \mathrm{R} \in \mathcal{R}'.\ \forall N_v \in \mathrm{V_R}.\ \forall N_s \in \mathrm{V_S}.$
        $\hat{x}_V(N_v, N_s) \leq x'_V(N_v, N_s)$ (23)
        $\forall \mathrm{R}^+ \in \mathcal{R}^+.\quad \hat{x}_\mathcal{R}(\mathrm{R}^+) = 1$ (24)
        $\forall \mathrm{R}^- \in \mathcal{R}^-.\quad \hat{x}_\mathcal{R}(\mathrm{R}^-) = 0$ (25)
    if $\hat{x_\mathcal{R}}(L[i]) = 1$ **then**
      set $t^s_{L[i]} \leftarrow \hat{t}^+_{L[i]}$ **and** $t^e_{L[i]} \leftarrow \hat{t}^-_{L[i]}$
      set $\mathcal{R}^+ \leftarrow \mathcal{R} \cup \{L[i]\}$
    else
      set $t^e_{L[i]} \leftarrow t^s_{L[i]} + \mathrm{d}_{L[i]}$
      set $\mathcal{R}^- \leftarrow \mathcal{R} \cup \{L[i]\}$
  return $(\hat{x}_\mathcal{R}, \hat{x}_V, \hat{x}_E, \hat{t}^+, \hat{t}^-)$

fix link allocations: even though the requests embedding period is fixed, link allocations are recomputed in each iteration.

We conclude our presentation of Algorithm $\text{c}\Sigma_{\text{A}}^{\text{G}}$ by showing that it is indeed polynomial. As the time points for starting and ending requests are fixed for all but one request, there are only $\mathcal{O}(\mathcal{R}'^2)$ many possibilities to map the temporally flexible request: simply place the point of time of the start between any two fixed time points and observe that the end can then be mapped on maximally $\mathcal{R}'$ many event points. As, given a fixed event order, the MIP reduces to an LP (of polynomial size), and as LPs can be solved in polynomial time using e.g., interior-point algorithms [14], Algorithm $\text{c}\Sigma_{\text{A}}^{\text{G}}$ can be implemented in polynomial time.

VI. Computational Evaluation

This section presents our computational evaluation of the algorithms to solve TVNEP, and focuses on the following questions: (1) How do the Δ, Σ and $\text{c}\Sigma$ formulations perform in comparison? (2) To which extent can we compute optimal solutions to the TVNEP using the $\text{c}\Sigma$ formulation? (3) How well does the Algorithm $\text{c}\Sigma_{\text{A}}^{\text{G}}$ perform in comparison? (4) What are the *benefits* of allowing temporal flexibility?

As underlying scenario we have chosen a synthetic workload on a data-center topology. The task is to solve the TVNEP for a day of work, represented by twenty requests spread over the day. We consider twenty-four of such workloads independently to allow for qualitative conclusions. As our main focus is how well our algorithms can cope with temporal flexibilities, in our plots (x-axis), we increment the temporal flexibilities (initially there are none) in steps of 30 "minutes" until each request is equipped with a temporal flexibility of 6 "hours". For each of the resulting $24 \times 11 = 264$ scenarios, we compute solutions using the Δ-, Σ-, $\text{c}\Sigma$- and the $\text{c}\Sigma_{\text{A}}^{\text{G}}$ algorithms with the access control objective.

After having presented the results under the access control objective, we consider the performance of the $\text{c}\Sigma$-Model under the three other objectives presented in Section IV-E.

A. Methodology

As substrate for our experiments we have chosen a directed 4×5 grid graph, with $|\text{V}_\text{S}| = 20$ nodes and $|\text{E}_\text{S}| = 62$ directed edges. Capacities on the substrate are set to be 3.5 for nodes and five for links. As request topologies, we use five node stars consisting of a single center and four surrounding nodes, such that either all links are directed towards the center or away from the center. This topology may represent a classical master-slave relationship, or a Virtual Cluster (e.g. [7]). The requests' required resources are chosen uniformly at random from the interval $[1, 2]$, such that with high probability only two nodes can be mapped on the same substrate node. We generate 20 requests from a Poisson arrival process with exponentially distributed inter-arrival time of 1 hour. The duration of requests is sampled from the (heavy-tailed) Weibull distribution with shape parameter two and scale parameter four, giving an expected duration of approximately 3.5 hours. As our focus in this work is to compare different continuous-time models, we fix node mappings a priori: for each virtual node we select uniformly at random a substrate node on which this node is to be embedded. Importantly however, the virtual links are not fixed beforehand; our algorithms therefore do not only need to find a subset of requests to embed, but furthermore need to decide when these requests are to be scheduled, compute the embeddings of the virtual links and guarantee that no substrate capacities are exceeded.

All experiments were conducted on Intel Xeon L5420@2,5 Ghz servers with 8 GB of RAM using Gurobi 5.60. Experiments were terminated after one hour of execution. All model and data files as well as the logs are available at [13].

B. Results

1) Access Control Objective: We start by comparing the different MIP formulations. Figure 3 depicts the runtime of the Δ-, Σ-, and $\text{c}\Sigma$-Model as a function of the time flexibility. As we terminate experiments after one hour of execution, Figure 4 shows the *objective gap* (the relative difference between upper and lower bound in the branch-and-bound process) after one hour. First, note that the Δ-Model is unable to generate solutions already for 90 minutes of flexibility, as only for one of the 24 scenarios, a solution is found. Considering the Σ- and $\text{c}\Sigma$-Model, both always yield feasible solutions. However, the runtime as well as the gaps of the $\text{c}\Sigma$-Model are on average one magnitude lower than for the Σ-Model. We therefore conclude that our optimizations presented in Section IV significantly reduce the runtime and improve the quality of solutions.

2) Performance of Algorithm $\text{c}\Sigma_{\text{A}}^{\text{G}}$*:* In Figure 7 the quality of solutions of the greedy Algorithm $\text{c}\Sigma_{\text{A}}^{\text{G}}$ is related to the (optimal) solutions determined by the $\text{c}\Sigma$-Model. While the median relative performance for 0 and 30 minutes of flexibility lies around 10%, the performance of $\text{c}\Sigma_{\text{A}}^{\text{G}}$ settles at around 5% for the rest of the temporal flexibilities. Even though in around 25% of the cases, Algorithm $\text{c}\Sigma_{\text{A}}^{\text{G}}$ yields solutions being off more than 10% compared to the (optimal) solution, the performance of $\text{c}\Sigma_{\text{A}}^{\text{G}}$ is rather surprising, since the optimal objectives increase nearly linearly (see Figure 9). We therefore conclude that Algorithm $\text{c}\Sigma_{\text{A}}^{\text{G}}$ scales well with increasing temporal flexibilities. Lastly, we state that the runtime of $\text{c}\Sigma_{\text{A}}^{\text{G}}$ to solve the $\text{c}\Sigma$ model with fixed time points for all but one request lies around 0.1 seconds per iteration. Algorithm $\text{c}\Sigma_{\text{A}}^{\text{G}}$ is therefore fast and generally produces high quality solutions.

3) Performance of $\text{c}\Sigma$*-Model under Different Objectives:* We will now consider the performance of the $\text{c}\Sigma$-Model under objectives not containing access control, but for maximizing earliness, the number of free nodes and the number of disabled links. To interpret the result in terms of the number of requests per flexibility, these are displayed in Figure 8. Figures 5 and 6 present the runtime and objective gap respectively. While the objective to disable seems to be the hardest of these, note that only at temporal flexibilities of 270 and 300 minutes, many scenarios did not yield optimal solutions within one hour of execution time. Furthermore, considering the runtime, optimal solutions for all three objectives can be computed within two

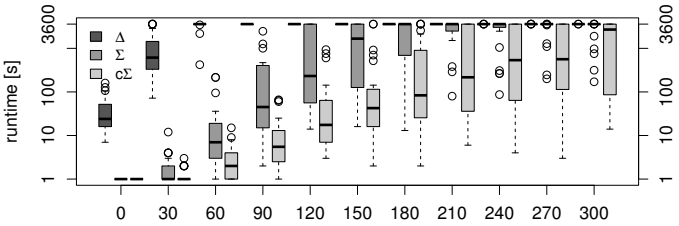

Figure 3. Runtime of MIP formulations as a function of the time flexibility. As computations were terminated after one hour, a runtime of 3600 implies, that no optimal solution has been found. Note the logarithmic y-axis.

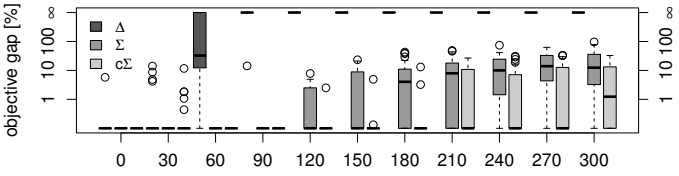

Figure 4. Objective gap of the formulations as a function of the time flexibility, if no optimal solution could be determined after one hour of execution. ∞ denotes that not a single solution was found. Note the logarithmic y-axis.

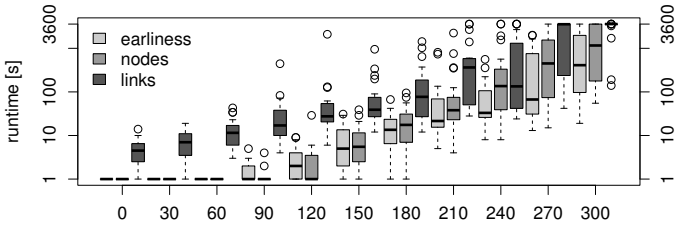

Figure 5. Runtime of the cΣ-Model under the objectives presented in Section IV-E. Note the logarithmic y-axis.

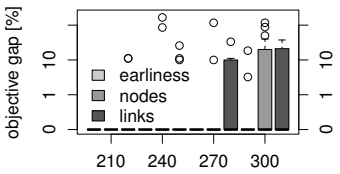

Figure 6. Gap of the cΣ-Model under the objectives presented in Section IV-E.

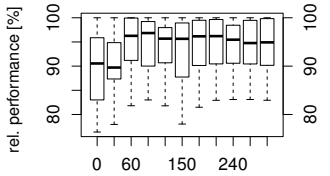

Figure 7. Relative performance of Algorithm $c\Sigma_A^G$ with respect to the solutions found by the cΣ-Model.

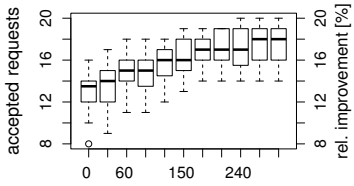

Figure 8. Number of requests embedded by the cΣ-Model.

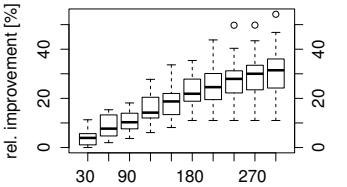

Figure 9. Relative improvement of the access control objective compared with the objective at flexibility 0.

minutes up to a temporal flexibility of 180 minutes.

VII. Related Work

For a general overview of the vision of cloud computing and network virtualization, the reader is referred to the surveys [16] and [17]. Overall, to the best of our knowledge, there is no previous work on the temporal VNet embedding problem. In the following, we review related literature in related fields.

a) Predictable Cloud Network Performance: Many papers have argued that not only computational and storage resources must be managed efficiently, but also networking matters (e.g., [3]). A nice overview of networking aspects in cloud computing is provided in [2]. While initially, mostly graph-based models were studied, see e.g., the SecondNet system [6], there is now a trend towards hose models known from wide-area networks, see e.g., the Oktopus system [7]. The algorithms presented in this paper are rather general and support all these models.

b) Time-varying Approaches: Until recently, most literature ignored the possibility that an application's network demands can vary over time [18], [19] in a predictable manner. Xie et al. [8] profiled several data-intensive MapReduce-style applications, and showed that many such applications exhibit predictable time-varying behavior at timescales on the order of tens of seconds. They proposed a Temporally-Interleaved Virtual Cluster (TIVC) abstraction, which allows the provider to admit multiple jobs that effectively time-share the same bandwidth. In the Cicada approach [2], the tenant starts with an initial placement of VMs and an estimated traffic matrix. The provider then profiles the application (similar to Proteus, but over longer timescales), with the aim of predicting the traffic matrix and how it varies diurnally. If the matrix appears predictable, the provider may propose to the tenant a time-varying or spatially-varying guarantee for future periods.

From an economical perspective, the idea to trade time flexibility with price has been studied in [20] from a scheduling complexity perspective. More generally, there are several interesting proposals for novel adaptive resource and spot market pricing schemes, e.g., [21], [22].

c) Embedding and Scheduling: The virtual network embedding problem has been studied intensively over the last years. The general formulation is NP-hard [23], and existing literature falls into four main categories: some works focus on *optimal solutions* on smaller scale environments (such as a router site), e.g., [24], others propose *approximation algorithms* (e.g., [12]) with provable quality guarantees or *heuristics* (e.g., [25]) which perform well, e.g., in simulations, and some works even propose to study environments which mitigate the computational complexities by imposing certain structures (e.g., [26]). Researchers have also pursued mathematical programming approaches already. For instance, Kumar et al. [27] describe an approach to solve a simpler Virtual Private Network tree computation problem for bandwidth provisioning. Even et al. [28] propose a general access control algorithm for many different routing and traffic models which selects and embeds only the requests of high benefit such that the overall benefit is maximized, but without exploiting placement or time flexibilities. Chowdhury et al. [12] present a MIP supporting flexible node placements for VNets, and propose a relaxation strategy to find solutions quickly.

Surprisingly, however, while a large body of literature exists on static embeddings, the important time-aspects for short-term embeddings have received much less attention. In this respect, the closest literature to ours are arguably the works on

the temporal routing and max flow problems which focus on links only (e.g., [29]). A limited form of temporal embedding support is also contained in the works by Zhang et al. [30] who also introduce a heuristic topology-aware embedding scheme, or in the context of VPN embeddings [28]; however, the opportunities of time flexibilities are not explored.

d) Other: Also in other fields with time-critical applications there exist mathematical programming solutions, such as the spatio-temporal composition of distributed multimedia objects [31], or more remotely, e.g., chemical production planning [32]. Indeed, several concepts introduced in the chemical production literature also apply to the VNet embedding problem [33]. However, the focus in chemical production planning lies on the sequential planning of production *processes*, where tasks can only be performed after the completion of predecessor tasks whose output is needed as input; time and specification flexibilities play only a secondary role. We are not aware of any results on more complex objective functions with time-dependent variables, whose state (e.g., resource allocations) and state differences needs to be tracked over time, as it is achieved with our approach.

VIII. Conclusion

As today's networks are becoming more virtualized, enabling a flexible service allocation and resource sharing, it is important that a predictable performance is ensured by resource isolation. Virtual networks can provide such guarantees. However, for an optimal resource allocation, VNets should be mapped and scheduled in a flexible and efficient manner.

Accordingly, this paper has introduced the TVNEP problem and presented the cΣ-Model to solve the TVNEP to optimality. Interestingly, despite the computational hardness of TVNEP, reasonably sized problem instances can be solved to optimality using our approach. We have additionally presented a greedy heuristic which is attractive if near optimal solutions are sufficient, and which could also be used in combination with the optimal algorithms, e.g., for allocating many smaller VNets while more rigorous optimizations are performed on the resource-intensive VNets (the "heavy-hitters").

IX. Acknowledgements

This work has been partially supported by the EU project BigFoot (FP7-ICT-317858).

References

[1] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: A scalable and flexible data center network," in *Proc. of the ACM SIGCOMM '09*, 2009.

[2] P. Dourish, "What we talk about when we talk about context," *Personal and Ubiquitous Computing*, vol. 8, no. 1, 2004.

[3] R. Nathuji, A. Kansal, and A. Ghaffarkhah, "Q-clouds: Managing performance interference effects for qos-aware clouds," in *Proc. of the EuroSys '10*, 2010.

[4] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, Apr. 2010.

[5] A. Haider, R. Potter, and A. Nakao, "Challenges in resource allocation in network virtualization," in *ITC Specialist Seminar '09*, vol. 18, 2009.

[6] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "SecondNet: A data center network virtualization architecture with bandwidth guarantees," in *Proc. of the CoNEXT '10*, 2010.

[7] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *Proc. of the ACM SIGCOMM '11*, 2011.

[8] A. Singla, A. Singh, K. Ramachandran, L. Xu, and Y. Zhang, "Proteus: A topology malleable data center network," in *Proc. of the ACM SIGCOMM HotNets Workshop '10*, 2010.

[9] L. Mai, E. Kalyvianaki, and P. Costa, "Exploiting time-malleability in cloud-based batch processing systems," in *Proceeding of the ACM SIGOPS LADIS Workshop '13*, 2013.

[10] S. Jain et al., "B4: experience with a globally-deployed software defined wan," in *Proc. of the ACM SIGCOMM '13*, 2013.

[11] A. Fischer, J. Botero, M. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 4, 2013.

[12] N. Chowdhury, M. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *Proc. of the IEEE INFOCOM '09*, 2009.

[13] M. Rost and S. Schmid, "Tech report and resources of simulation," in *http://www.net.t-labs.tu-berlin.de/ stefan/tvnep.html*, Oct 2013.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

[15] S. Seetharaman, "Energy conservation in multi-tenant networks through power virtualization," in *Proc. of the USENIX HotPower '10*. Berkeley, CA, USA: USENIX Association, 2010.

[16] M. Armbrust et al., "Above the clouds: A berkeley view of cloud computing," in *UC Berkeley TR*, vol. 28, 2009.

[17] A. Belbekkouche, M. M. Hasan, and A. Karmouch, "Resource discovery and allocation in network virtualization," *Communications Surveys Tutorials, IEEE*, vol. 14, no. 4, Fourth 2012.

[18] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. of the ACM SIGCOMM IMC '10*, 2010.

[19] Y. Chen, S. Alspaugh, and R. Katz, "Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads," *Proc. VLDB Endow.*, vol. 5, no. 12, 2012.

[20] T. A. Henzinger, A. V. Singh, V. Singh, T. Wies, and D. Zufferey, "A marketplace for cloud resources," in *Proc. of the ACM EMSOFT '10*, 2010.

[21] V. Abhishek, I. A. Kash, and P. Key, "Fixed and market pricing for cloud services," in *Proc. IEEE INFOCOMM '12 WKSHPS Workshop*, 2012.

[22] R. Pal and P. Hui, "Economic models for cloud service markets," in *Proc. of the ICDCN '12*. Springer-Verlag, 2012.

[23] D. Andersen, "Theoretical approaches to node assignment," Carnegie Mellon University, Tech. Rep., 2002.

[24] G. Schaffrath, S. Schmid, and A. Feldmann, "Optimizing long-lived cloudnets with migrations," in *Proc. of IEEE UCC '12*, 2012.

[25] J. Fan and M. H. Ammar, "Dynamic topology configuration in service overlay networks: A study of reconfiguration policies," in *Proc. of the IEEE INFOCOM '06*, 2006.

[26] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, 2008.

[27] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener, "Algorithms for provisioning virtual private networks in the hose model," *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, 2002.

[28] G. Even, M. Medina, G. Schaffrath, and S. Schmid, "Competitive and deterministic embeddings of virtual networks," in *Proc. of the ICDCN '12*. Springer-Verlag, 2012.

[29] E. Köhler and M. Skutella, "Flows over time with load-dependent transit times," in *Proc. of the ACM-SIAM SODA '02*. SIAM, 2002.

[30] S. Zhang, Z. Qian, J. Wu, and S. Lu, "An opportunistic resource sharing and topology-aware mapping framework for virtual networks," in *Proc. of the IEEE INFOCOM '12*, 2012.

[31] H.-J. Lin, "Interval algebra for spatio-temporal composition of distributed multimedia objects," in *Proc. International Conference on Parallel and Distributed Systems (ICPADS)*, 1998.

[32] X. Lin and C. Floudas, "Design, synthesis and scheduling of multipurpose batch plants via an effective continuous-time formulation," *Computers & Chemical Engineering*, vol. 25, no. 4, 2001.

[33] C. A. Floudas and X. Lin, "Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review," *Computers & Chemical Engineering*, vol. 28, no. 11, 2004.