# Using Optimized Virtual Network Embedding for Network Dimensioning

Johannes Inführ*, David Stezenbach†, Matthias Hartmann †, Kurt Tutschku† and Günther R. Raidl*

*Vienna University of Technology, Institute of Computer Graphics and Algorithms
†University of Vienna, Chair of Future Communication (Endowed by A1 Telekom Austria AG)
{david.stezenbach,matthias.hartmann,kurt.tutschku}@univie.ac.at, {infuehr,raidl}@ads.tuwien.ac.at

***Abstract*—Virtual Network Embedding will be one of the key concepts of the Future Internet. For an ISP it is important to know how many additional Virtual Networks (VNs) of a specific application (e.g. web, streaming, P2P, and VoIP) are mappable into the current resource substrate with a certain probability. In this work we calculate this probability with our embedding algorithm which enables us to consider side effects based on remapping of VNs (e.g. due to reduced link delay). Our results show that minimal extra resources can significantly increase embedding probability of additional VNs.**

***Index Terms*—Virtual Network Embedding, Network Dimensioning, Service Availability, Network Optimization**

## I. Introduction

A current trend in research and productive operation is network virtualization. Furthermore, rapid configuration and deployment of Virtual Networks (VNs) (e.g. Gush [1], Teagle [2], RSpec [3]), as well as fast flow based traffic engineering (e.g. OpenFlow [4], Junos SDK [5]) are currently hot topics. In conjunction with federation, which is the willingness and capability of different providers to share network resources, this leads to a higher degree of freedom in the selection of available resources and the embedding of VNs into the substrate. This allows the Internet Service Providers (ISPs) to support the requirements of applications better than it is currently possible. The VNs are expected to be application specific with own naming schemes, topologies, custom routing algorithms and even specialized resource management. Thus, different types of applications will utilize different VNs in order to handle their traffic in an appropriate and near optimal way.

In order to enable the network to support highly dynamic application requirements, flexible resources need to be available. This flexibility can be achieved by leasing virtualized resources. Based on these resources VNs for different applications are built. These VNs will operate on a much shorter time scale than todays Virtual Private Networks (VPNs), but will last longer than flow based approaches like RSVP/IntServ [6] or DiffServ [7], [8]. This heterogeneous pool of resources handled by different providers requires an adequate description which we discussed in [9]. Furthermore, finding an optimal mapping of the VNs into the substrate is a NP-hard problem, as is finding a valid mapping in the first place. Taking into account that federation enlarges the resource pool this results in an optimization problem which is even harder to solve. We presented this Virtual Network Mapping Problem with Delay, Routing and Location constraints (VNMP-DRL) in detail in [10], which we will continue to improve in order to be applicable in a real-world concept to find a near optimal solution in a very short time frame.

The "OptFI" project has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT10-027. EuroNF funded this work through WP IA.4.1 on "Mobility of Ph.D. Students"

In this work we evaluated the capability of our mapping algorithm to be used for network dimensioning decisions and resource leasing strategies. We opine that network dimensioning should be performed by the same algorithm as the mapping process. This considers side effects resulting from remapped VN (e.g. caused by path delay reduction) which could bias dimensioning based on shortest path flow mapping. The network dimensioning is based on the probability that a new VN can be mapped into the current substrate under the given resource requirements (e.g. bandwidth, delay, routing capacity) and the current load caused by other VN. During the mapping process we identified the bottlenecks of the substrate network which prohibit the further embedding of VNs. Thus, we are able to point out at where a minimum of extra resources can increase the probability of an additional VN embedding possibility. Identifying these bottlenecks in advance enable the ISPs to lease and preconfigure extra resources on these spots in order to reduce the embedding time of new VNs. Afterwards, we show that these selective improvements of the network lead to significantly higher service availability for VNs of a specific type. This offers the ISPs to either dimension their networks for a specific VNs type in advance or for a general embedding probability.

The remainder of this paper is structured as follows. We give an overview on related work in Section II. The four different application VN are described in Section III. Followed, be the calculation of the blocking probability for the VNMP-DRL in Section IV. The results of our evaluations are presented in Section V. Finally, we conclude our work in Section VI.

## II. Related Work

A network topology must be able to cope with the offered traffic and avoid packet loss and delay. As the actual amount of data that is transported during normal operation can only be estimated, different mechanisms can be used to ensure adequate quality of service by avoiding or minimizing link overload in the network. Up to a certain level, traffic engineering (TE) can be used to optimize the resource-usage of flows in the network [11]. This allows to carry more traffic through a given network topology, but the optimization is a difficult problem that requires carefully select objective functions [12]. Another mechanism, which can also be used in addition to TE, is admission control (AC). It was proposed for the Internet in [13]. Here, the network load is analyzed before new flows are allowed in the network in order to avoid overload situations. Both mechanisms are rather complex and require constant monitoring of the network. A much simpler approach is to use capacity overprovisioning (CO). Here, the network is overdimensioned, which makes overload in unexpected scenarios very unlikely. It is often stated that CO has high initial costs in comparison to other methods, but as the network usually must also provide backup capacity for unexpected failure scenarios, the bandwidth requirements for CO are even similar than those of AC [14]. To determine the best capacities for links and routers is not easy, even when accurate information about traffic patterns in a network are provided. A method to find network elements which might have insufficient capacity is implemented in the Resilyzer framework [15], [16]. It can analyze all network failure scenarios with given probability and determine for example the link overload probability, which can be used for CO.

In this paper we provide a new method to determine such bottlenecks that can be alleviated by CO, which is not based on failure probabilities but on the blocking probability of VN embedding optimization. There are many approaches to provide optimal or at least good embeddings of VNs into a substrate. [17] introduces a framework to compare the performance of the major proposed VNE algorithms. To the best of our knowledge, no other previous work examined the use of a Virtual Network Embedding (VNE) algorithm for finding bottlenecks in the substrate.

## III. Network Traffic Model

In this work, we used the same network traffic model as the VNMP-DRL uses. This section will give a short overview of this problem and its associated test instances. Interested readers are referred to [10] for more detailed information. A VNMP-DRL instance consists of two graphs. The first one is a directed multigraph which represents the substrate. The nodes of the substrate graph have associated CPU and routing capacities. The available CPU capacity determines how much processing power is available for the VNs, for instance for implementing custom routing protocols. The routing capacity determines how much bandwidth can traverse a router and is usually less than the total connected bandwidth. Arcs representing links have bandwidth capacities and incur some

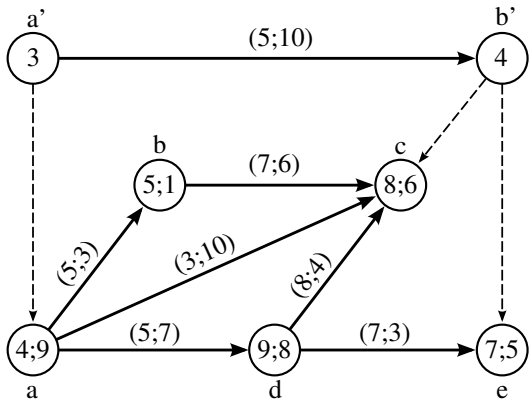


Fig. 1: Example of a VNMP-DRL instance.

delay. The second graph is the VN graph. Its disconnected components specify all the VNs that have to be mapped onto the substrate graph. Nodes have associated CPU requirements and arcs representing desired connections are specified with their needed bandwidths and limits on the maximum allowed delays. As a third component there is a mapping which specifies on which substrate nodes the VN nodes are allowed to be mapped. The goal is to find a feasible mapping of the VN nodes to substrate nodes and an implementing path in the substrate for each VN arc so that all constraints are satisfied.

Figure 1 shows a small VNMP-DRL instance. It contains the VN graph consisting of one VN with two nodes ($a'$ and $b'$), the substrate graph ($a$ to $e$) and the allowed mapping from VN to substrate nodes (dashed lines). The node labels define the CPU requirement for VN nodes and the available CPU and routing capacity for substrate nodes. The arc labels denote bandwidths and delays. Note that in this example, $b'$ actually cannot be mapped to $c$, even though $c$ offers enough CPU and routing capacity. This is because there is no path from $a$ to $c$ that satisfies the constraints of the virtual connection between $a'$ and $b'$. Node $b$ cannot be used, because its routing capacity of 1 cannot support the required bandwidth of 5. The direct connection from $a$ to $c$ does not offer enough bandwidth and the path over $d$ exceeds the delay limit. The only possible solution of this instance is to map $a'$ to $a$ and $b'$ to $e$ and implement the connection between $a'$ and $b'$ by using the path from $a$ to $d$ and then to $e$.

The VNMP-DRL instances used in this work [18] are based on the Internet network maps produced by the rocketfuel [19], scan [20] and lucent [21] projects. Subgraphs of those internet network maps were extracted to form the substrate graphs. The VN graphs consist of instances of four different VN types, which were chosen to cover the more important use cases:

Stream This type models (video-)streaming services and has tree structure. The root node represents the content provider which broadcasts content consisting of multiple channels. Intermediate nodes in the tree split this stream and forward only the channels that are actually watched by the customers connected to the leaf nodes of the distribution tree. This is

an example of an application that can benefit from custom routing protocols. The stream VN type has high bandwidth and processing power demands but no delay constraints.

Web This type models the typical web usage and has star structure. The center represents the web-server which serves the connected customers. This VN type has low bandwidth and processing power demands but stringent delay constraints.

P2P This type represents peer-to-peer (P2P) networks and is based on a highly connected graph. It has high bandwidth and medium processing power demands but no delay constraints.

VoIP This type models voice over IP (VoIP) networks and is again based on a highly connected graph. It has medium bandwidth and processing power demands. Delay limits are chosen fairly limiting to allow for voice communication.

## IV. Methodology

The basic idea of the conducted experiments is to show that Virtual Network Provider (VNP) can extract valuable information from VN that they could not accept in order to improve the substrate. By improvement we mean that the probability that an additional VN can be embedded into the substrate (the embedding probability $p_{\mathrm{em}}$) is increased, and that more VNs can be embedded into the substrate before $p_{\mathrm{em}}$ reaches zero.

To show this, we perform two experiments. We start with the VNMP-DRL instances available at [18]. For those instances, we determine $p_{\mathrm{em}}$ separately for each VN type because we want to show the influence of different VN types on $p_{\mathrm{em}}$. The embedding probability of an instance is determined by adding a VN of a specific type and checking whether this new problem is solvable or not. This is repeated $n_{\mathrm{S}}$ times and $p_{\mathrm{em}}$ is calculated as the fraction of solvable cases in relation to the number of tested ones. In this work, we use $n_{\mathrm{S}} = 50$ as tradeoff between required runtime and confidence in the calculated $p_{\mathrm{em}}$. We continue this process with the first found solvable problem (i.e. the added VN could be successfully embedded in the substrate) in a depth-first fashion until $p_{\mathrm{em}}$ reaches zero or we have added ten additional VNs. Then the procedure tracks back to an instance with the least amount of added VNs that has not had its $p_{\mathrm{em}}$ determined. The motivation for this traversal order is that we want to see how $p_{\mathrm{em}}$ develops while adding additional VNs but also cover a reasonable fraction of the search space of instances reachable from the initial problem. We chose the upper limit of ten additional VNs because as more VNs are added, the resource bottlenecks in the substrate depend more and more on the exact configuration of the added VNs and not on the initial situation in the substrate and therefore become less relevant when we want to determine where we want to add additional resources to the substrate. Additionally, we used a limit of 101 $p_{\mathrm{em}}$ evaluations, which allows the cutoff point of ten added VNs to be reached ten times. For the whole procedure (which we will call extension procedure from now on) we set a time limit of one day and we executed it for each of the four VN types used for the VNMP-DRL.

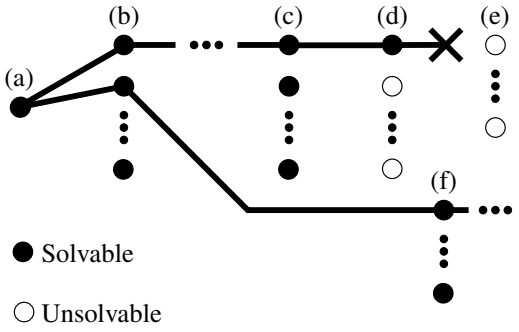



Fig. 2: Example of an extension procedure execution.

Figure 2 shows an example execution of the extension procedure. It starts with problem instance $(a)$, an instance from the instance set. It then determines $p_{\mathrm{em}}$ for this instance by repeatedly extending $(a)$ with one random VN (generated as in [10]) and checking whether the new instances $(b)$ are solvable or not. In the illustrated case, every newly generated instance was solvable, we continue with the first found instance and repeat this process until we reach $c$. For $c$, we determine a $p_{\mathrm{em}}$ of 2%, because only one of 50 instances is solvable. We continue with the only found solvable instance $(d)$. Node $(d)$ has a $p_{\mathrm{em}}$ of 0%, so there are no instances with which to continue; we track back to an instance with the least amount of added VNs that has not had its $p_{\mathrm{em}}$ determined, in this case the second solvable instance created from $(a)$. The extension procedure then continues with $(f)$ and so on until one of the mentioned limits is reached or no more instances remain.

During the extension procedure, a lot of unsolvable instances (i.e. VN configurations that cannot be embedded into the substrate) are created. For the second experiment, we add resources to the substrates of the starting instances based on the reasons why those instances from the first run of the extension procedure were unsolvable and then execute the extension procedure again.

The following subsections describe in detail how we determined the solvability or unsolvability of a problem instance during the calculation of $p_{\mathrm{em}}$, how the reasons for unsolvability were extracted and how they were translated into substrate changes.

### A. Proving Unsolvability and Extracting Reasons

During the execution of the extension procedure, we want to collect reasons (i.e. location and amount of missing resources) why some VN configurations could not be mapped into the substrate so that we can derive changes to the substrate from those reasons. To get the unsolvability reasons, we solve a modified version of the Integer Linear Program (ILP) presented in [10]. It calculates the cheapest changes to the substrate (i.e. added resources), so that a specific VN configuration can be embedded. We used different resource

costs for each type of resource, because for instance changing the CPU capacity of a substrate node will generally be cheaper than decreasing the delay of a substrate connection. We will now present the used ILP in detail.

The directed multigraph $G = (V, A)$ with node set $V$ and arc set $A$ represents the substrate network. The available CPU power of a substrate node is denoted with $c_i \in \mathbb{N}^+,\ \forall i \in V$, its routing capacity with $r_i \in \mathbb{N}^+,\ \forall i \in V$. The delay of a substrate arc is denoted by $d_e \in \mathbb{N}^+,\ \forall e \in A$, its available bandwidth with $b_e \in \mathbb{N}^+,\ \forall e \in A$. The components of the VN graph $G' = (V', A')$ are the VNs that have to be embedded into the substrate. The constant $c_k \in \mathbb{N}^+,\ \forall k \in V'$ determines the required CPU power of a virtual node. The required bandwidth by a virtual connection is defined by $b_f \in \mathbb{N}^+,\ \forall f \in A'$ and the maximum allowed delay by $d_f \in \mathbb{N}^+, \forall f \in A'$. The set $M \subseteq V' \times V$ defines the allowed mappings between virtual and substrate nodes. The functions $s : A \cup A' \to V \cup V'$ and $t : A \cup A' \to V \cup V'$ associate each arc of $G$ and $G'$ with their source and target nodes, respectively. The coefficients $C^{\text{CPU}}$ $C^{\text{RC}}$ $C^{\text{BW}}$ $C^{\text{DL}}$ define the cost of adding one unit of CPU or routing capacity to a substrate node, a unit of bandwidth to a substrate arc or removing a unit of delay from a substrate arc. In this work $C^{\text{CPU}} = C^{\text{RC}} = 1$, $C^{\text{BW}} = 5$ and $C^{\text{DL}} = 20$.

The ILP utilizes the decision variables $x_{ki} \in \{0, 1\},\ \forall k \in V',\ \forall i \in V$ to indicate where the virtual nodes are located in the substrate graph and $y_e^f \in \{0, 1\},\ \forall f \in A',\ \forall e \in A$ to indicate if a virtual connection is implemented by using a substrate connection. The decision variable $z_i^f \in \{0, 1\},\ \forall f \in A',\ \forall i \in V$ indicates that a substrate node is used to route a virtual connection. The variables $a_i^{\text{CPU}},\ \forall i \in V$ represent the added CPU capacity to a substrate node, $a_i^{\text{RC}},\ \forall i \in V$ the added routing capacity. $a_e^{\text{BW}},\ \forall e \in A$ represents the added bandwidth to a substrate arc and $a_e^{\text{DL}},\ \forall e \in A$ the removed delay. Note that $a_e^{\text{DL}}$ is always non negative. The auxiliary variable $d_e^f,\ \forall e \in A,\ \forall i \in V$ represents the delay a substrate arc has when used to implement a virtual arc and is required due to technical reasons.

The complete ILP is defined by inequalities (1)–(18).

$$\min \quad \sum_{i \in V} (C^{\text{CPU}} a_i^{\text{CPU}} + C^{\text{RC}} a_i^{\text{RC}}) + \sum_{e \in A} (C^{\text{BW}} a_e^{\text{BW}} + C^{\text{DL}} a_e^{\text{DL}}) \quad (1)$$

$$\sum_{(k,i) \in M} x_{ki} = 1 \qquad \forall k \in V' \quad (2)$$

$$\sum_{e \in A | t(e) = i} y_e^f + x_{s(f)i} - \sum_{e \in A | s(e) = i} y_e^f - x_{t(f)i} = 0 \qquad \forall i \in V,\ \forall f \in A' \quad (3)$$

$$\sum_{e \in A | t(e) = i} y_e^f + x_{s(f)i} \leq z_i^f \qquad \forall i \in V,\ \forall f \in A' \quad (4)$$

$$\sum_{(k,i) \in M} c_k x_{ki} \leq c_i + a_i^{\text{CPU}} \qquad \forall i \in V \quad (5)$$

$$\sum_{f \in A'} b_f z_i^f \leq r_i + a_i^{\text{RC}} \qquad \forall i \in V \quad (6)$$

$$\sum_{f \in A'} b_f y_e^f \leq b_e + a_e^{\text{BW}} \qquad \forall e \in A \quad (7)$$

$$d_e y_e^f - a_e^{\text{DL}} \leq d_e^f \qquad \forall e \in A,\ \forall f \in A' \quad (8)$$

$$a_e^{\text{DL}} \leq d_e - 1 \qquad \forall e \in A,\ \forall f \in A' \quad (9)$$

$$\sum_{e \in A} d_e^f \leq d_f \qquad \forall f \in A' \quad (10)$$

$$x_{ki} \in \{0, 1\} \qquad \forall (k, i) \in M \quad (11)$$

$$y_e^f \in \{0, 1\} \qquad \forall e \in A,\ \forall f \in A' \quad (12)$$

$$z_i^f \in \{0, 1\} \qquad \forall i \in V,\ \forall f \in A' \quad (13)$$

$$a_i^{\text{CPU}} \in \mathbb{R}_0^+ \qquad \forall i \in V \quad (14)$$

$$a_i^{\text{RC}} \in \mathbb{R}_0^+ \qquad \forall i \in V \quad (15)$$

$$a_e^{\text{BW}} \in \mathbb{R}_0^+ \qquad \forall e \in A \quad (16)$$

$$a_e^{\text{DL}} \in \mathbb{R}_0^+ \qquad \forall e \in A \quad (17)$$

$$d_e^f \in \mathbb{R}_0^+ \qquad \forall e \in A,\ \forall f \in A' \quad (18)$$

The objective is given by (1), the total cost of added resources is to be minimized. Equalities (2) ensure that each virtual node is mapped to exactly one substrate node, subject to the mapping constraints. The flow conservation constraints (3) make sure that for each virtual connection there is a connected path in the substrate. Linking constraints (4) ensure that variables $z_i^f$ are equal to one when the corresponding node is used to route the traffic of a particular virtual connection. Inequalities (5)–(7) ensure that the solution is valid with regard to CPU, routing capacity and bandwidth constraints while also considering added resources. Incorporating changes to the substrate delay is not as straight forward. Inequalities (8) together with the domain of $d_e^f$ ensure that $d_e^f$ is either zero if substrate arc $e$ is not used by virtual arc $f$ (i.e. $y_e^f$ is zero) or has the correct delay value (i.e. defined delay minus delay reduction) if the substrate arc is used. Inequalities (9) make certain that even after reduction, the delay of a substrate connection is at least 1. Inequalities (10) ensure that any solution is valid with regards to the delay constraints. Equations (11)–(18) define the domains of the used variables. Note that the model only includes integrality constraints for $x_{ki},\ y_e^f$ and $z_i^f$ (11)–(13), the already covered constraints together with the objective function cause $a_i^{\text{CPU}}$, $a_i^{\text{RC}}$, $a_e^{\text{BW}}$ and $a_e^{\text{DL}}$ to be integral as well. Every non-zero $a$ variable in an optimal solution counts as one failure reason because a resource had to be added in order to embed all VNs.

### *B. Reacting to Failure Reasons*

Executing the extension procedure for one instance typically creates $\approx 3000$ unsolvable VN configurations. With the help of the ILP formulation from the previous subsection, $\approx 8000$ reasons for unsolvability can be extracted. These have to be condensed into concrete changes for the substrate. The following lists the five parameters we used for calculating the changes to the substrate:

- $f$ Function that calculates a descriptive value from a set of missing resources, e.g. mean or max
- $s$ Scaling factor for the result of $f$ to calculate the added amount of resources
- $r$ How often resources have to be missing at a specific location in the substrate in relation to the maximum

number of reported missing resources in order for this location to receive additional resources

$n$ Maximum number of resource changes in the substrate (per resource type)

Our aim is to add more resources to the most critical parts of the substrate network. Critical parts are those that are frequently reported as having too few resources of a specific type available. Let this report count be called $m$, and the highest count $m_{\max}$. Note that the amount of missing resources that is reported is not yet relevant. So, for routing capacity, bandwidth, CPU power and delay separately, we regard substrate nodes (in case of routing capacity and CPU power) and arcs (for bandwidth and delay) in descending order of $m$. All locations with $m \geq r \cdot m_{\max}$, but at most $n$, will receive additional resources. These cutoff rules ensure that we do not add resources to too many locations in the substrate (which would not be economical) and also not to locations which only rarely miss resources. The amount of additional resources is determined by applying $f$ to the reported amounts of missing resources at the selected locations and multiplying the result by $s$. In this work, we used $f = mean$, $s = 3$, $r = 0.3$, $n = 5$.

## V. Results

Each computational experiment reported in this section has been performed on one core of an Intel Xeon E5540 multi-core system with 2.53 GHz and 3 GB RAM per core. The instances to which we applied the extension procedure are available at [18]. We exclusively used the instances with 80% of the maximum number of VNs so that there would still be some resources left for further VNs. Due to the computational demand of the extension procedure, we only used the first two instances (out of ten) for each topology source of the substrate graph and each size (i.e. 80 instances in total) for the extension procedure. CPLEX 12.4 [22] was used to solve the presented ILP model in single threaded mode and with a time-limit of 300 seconds. For instances with a substrate size of 100 nodes we used a time-limit of 500 seconds. Additionally, a memory limit of 5 GB was set. If the optimal solution to the ILP was not found within the time-limit, we used the best found feasible solution if the gap to the optimal solution was smaller than 95%. Preliminary runs showed that for larger substrates the feasible solution reported by CPLEX sometimes was the result of heuristics CPLEX runs before it actually starts to solve the ILP. For this problem, the solutions generated in this way add a lot of resources to almost all nodes and links in the substrate and are therefore not helpful for finding the real bottlenecks in the substrate. The gap limit ensures that we do not use those solutions.

### A. VNMP-DRL Instance Properties

In this section we show the properties of the 80 VNMP-DRL instances used as base for the performed experiments. Table I shows the number of nodes and arcs and Table II the number of VNs contained in the instances. Note that while the number of nodes and arcs contained in the instances rises with the instance size, the number of VNs does not (after size 20). This is because the VNs grow in size as the substrate grows. P2P and VoIP VNs are fewer than the other VN types because they require more resources individually. We tested only on 10 instances of size 100 because the instance set contains fewer instances of this size.

TABLE I: Number of used VNMP-DRL instances per size (#) and the average number of substrate arcs ($A$), VN nodes ($V'$) and VN arcs ($A'$) of those instances.

| Size | # | $\overline{\lvert A \rvert}$ | $\overline{\lvert V' \rvert}$ | $\overline{\lvert A' \rvert}$ |
|---|---|---|---|---|
| 20 | 14 | 53.7 | 118.2 | 146.0 |
| 30 | 14 | 95.4 | 215.0 | 264.0 |
| 40 | 14 | 126.1 | 275.4 | 382.0 |
| 50 | 14 | 172.1 | 308.0 | 523.9 |
| 70 | 14 | 259.4 | 478.7 | 835.6 |
| 100 | 10 | 399.0 | 648.0 | 1276.9 |

TABLE II: Average number of Stream, Web, P2P and VoIP VNs contained in the used VNMP-DRL instances.

| Size | $\overline{\text{Stream}}$ | $\overline{\text{Web}}$ | $\overline{\text{P2P}}$ | $\overline{\text{VoIP}}$ |
|---|---|---|---|---|
| 20 | 6.6 | 8.5 | 4.4 | 4.1 |
| 30 | 13.1 | 14.6 | 7.6 | 7.8 |
| 40 | 16.4 | 18.5 | 6.9 | 7.1 |
| 50 | 14.4 | 16.5 | 7.5 | 6.4 |
| 70 | 18.1 | 17.9 | 6.1 | 7.6 |
| 100 | 14.0 | 15.8 | 6.8 | 9.1 |

### B. Extension Procedure

After we executed the extension procedure for all 80 instances and for each of the four VN types, the extracted failure reasons were distributed as shown in Table III.

It can be seen that in general missing routing capacities are prevalent, no matter which type the additionally added VNs are. For stream VNs, the fraction of missing routing capacities is reduced while CPU capacities are missing more often with increasing substrate size. Normally, one would expect the development to be the other way around, because routing capacities are needed at multiple nodes in the substrate network to implement a virtual connection, while CPU capacities are only needed at the start and the end of a virtual connection. When the substrate grows, so do the average lengths of the implementations of virtual connections and the total required routing capacity increases while the CPU capacity stays the same. This would cause the probability of missing routing capacity to increase. The reason why missing CPU resources become more prominent is that the nodes of stream VNs require a lot of CPU capacity (because they have to split and distribute video streams). The VN sizes grow proportional to the substrate sizes, therefore the stream VNs for larger substrates can contain more nodes which perform video stream splitting which in turn requires more CPU resources. In relation to the missing CPU and routing capacities, the substrate link bandwidths only play a subsidiary

TABLE III: Fraction of failure reasons of a specific type (missing CPU- (C) or routing- (R) capacity, missing bandwidth (B) or too much delay (D)) during the first run of the extension procedure for each VN type in relation to the total number of found failure reasons in percent. (May not add up to exactly 100% due to rounding.)

| | Stream | | | | Web | | | | P2P | | | | VoIP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Size** | **C** | **R** | **B** | **D** | **C** | **R** | **B** | **D** | **C** | **R** | **B** | **D** | **C** | **R** | **B** | **D** |
| 20 | 39.9 | 56.0 | 4.1 | 0.0 | 12.5 | 66.2 | 0.0 | 21.3 | 11.9 | 80.3 | 7.8 | 0.0 | 11.5 | 88.0 | 0.2 | 0.2 |
| 30 | 35.5 | 43.7 | 20.7 | 0.0 | 10.6 | 40.8 | 12.3 | 36.3 | 5.1 | 53.3 | 41.6 | 0.0 | 4.3 | 48.1 | 40.3 | 7.3 |
| 40 | 42.9 | 46.1 | 11.0 | 0.0 | 8.9 | 37.5 | 9.1 | 44.4 | 13.3 | 58.8 | 27.9 | 0.0 | 16.2 | 42.2 | 12.9 | 28.7 |
| 50 | 39.5 | 38.6 | 21.9 | 0.0 | 6.2 | 38.0 | 12.7 | 43.1 | 1.8 | 78.7 | 19.0 | 0.5 | 4.6 | 45.5 | 31.6 | 18.4 |
| 70 | 53.7 | 33.4 | 12.8 | 0.0 | 9.2 | 28.0 | 13.3 | 49.4 | 6.0 | 61.7 | 31.7 | 0.6 | 5.6 | 58.7 | 34.0 | 1.7 |
| 100 | 48.9 | 36.5 | 14.6 | 0.0 | 4.0 | 42.6 | 3.3 | 50.1 | 5.2 | 88.8 | 6.1 | 0.0 | 19.1 | 74.1 | 5.1 | 1.6 |

role and no link was found to have too much delay, which is not surprising since the virtual connections in stream VNs are not delay constrained.

This is not true for web VNs which are heavily delay constrained. For substrates of size 100, more than half of all found failure reasons are too high delay on some substrate arc. This fraction rises with increasing substrate size because the average length of virtual arc implementations and therefore the total accumulated delay grows.

When embedding additional P2P VNs into the substrate, missing bandwidths on substrate links become a significant issue for the first time, even though missing routing capacities still dominate. Again, this is because the (already high) bandwidth requirements of P2P VNs require a lot of routing capacity in the substrate nodes and depending on the particular substrate configuration one resource is more limiting than the other. Also note that a bias against bandwidth changes exists, since adding one unit of bandwidth costs five times more than a unit of routing capacity. That means that if an additional VN can be added either by adding four units of routing capacity or adding one unit of bandwidth, four units of routing capacity are reported missing because this is the cheaper solution. However, this effect can also be witnessed the other way around with this VN type. Even though P2P VNs are not delay constrained, for sizes 50 and 70 delays were reported missing. This can happen because the initial instances contain all four VN types, i.e. also some with delay constraints. By reducing the delay somewhere, the virtual connections with delay requirements can be implemented by using a previously impossible substrate path which frees up resources for new VNs where they are needed. This can be the cheaper solution, even though decreasing link delay by one unit costs twenty times more than adding a unit of routing capacity for instance.

For VoIP VNs, missing bandwidths are again prominent (after missing routing capacities). Also link delays play some role, but since VoIP VNs are not as delay constrained as web VNs it is not surprising to see that delays are less often a failure reason when compared to web VNs. Unfortunately we can offer no conclusive reason as to why the fraction of delay reasons fluctuates so much for VoIP VNs. Larger instances reduce the probability that the presented ILP can be solved to optimality, so one possible explanation could be that feasible solutions which only change other resources are predominantly found. Due to space constraints we cannot go into more detail with regards to the performance of the ILP, but in a nutshell, it works well up to substrate sizes of 50, i.e. more than 99% of executions yield useful results (a valid solution within the gap limit). For sizes 70 and 100, there is a split between stream and web VNs on the one hand and P2P and VoIP VNs on the other. For stream and web VNs, we still get useful results more than 90% of the time. For P2P and VoIP VNs, this is reduced to 85% for instances of size 70 and even 45% for size 100.

Based on the reported missing ressources for each instance, we added resources to each VNMP-DRL instance as outlined in Subsection IV-B. Table IV shows the change to the available resources in the substrate because of the added resources.

It can be seen that only a very small amount of resources was added to the substrate, as was our goal. Generally, the total available amount of resources was increased by less than one percent. The most notable exception to this are the delays when adding Web or VoIP VNs. There are three factors that work together to cause this. First of all, web and VoIP VNs are delay constrained, so even though delays are not very often reported to be too high, the reported magnitude *is* high. Secondly, the astute reader will have noted that $s$ is also applied when calculating delay changes for uniformity reasons, even though it is not strictly necessary. If the delay of a substrate connection is reduced all future VNs will benefit, at least in our simplified model where substrate link delays are independent of link load. The same is not true for the other resource types, since they are used up by additional VNs, which is the reason why we add more resources than are reported missing. The third contributing factor to the large delay changes is the fact that the sum of all delays in the substrate is less than the sum of the other resource types, so changing the delay by some fixed amount will be a larger relative change than for the other resources.

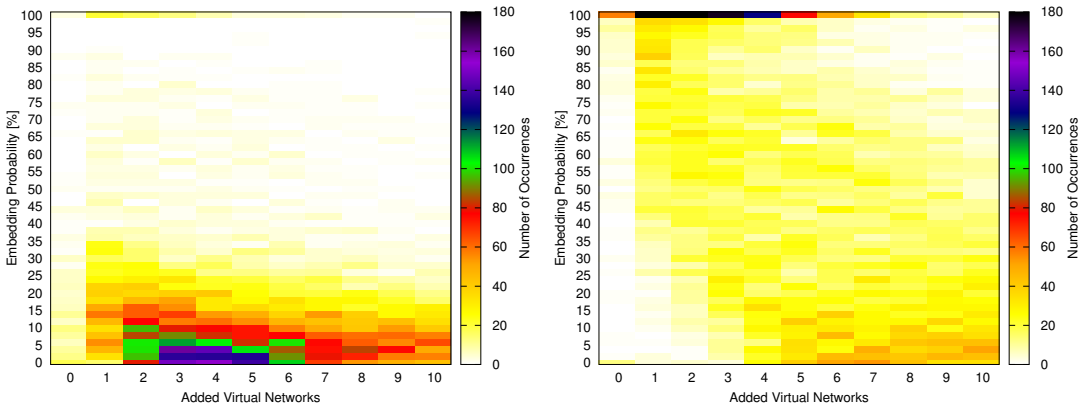
The following subsection will answer the question whether the resources added to the substrate could influence $p_{\mathrm{em}}$ in a meaningful way.

### C. Change to $p_{\mathrm{em}}$

The complete development of $p_{\mathrm{em}}$ during both runs of the extension procedure can be seen in Figure 3. It shows how

TABLE IV: Relative change of the available resources in the substrate due to the found failure reasons in percent.

| | Stream | | | | Web | | | | P2P | | | | VoIP | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Size** | **C** | **R** | **B** | **D** | **C** | **R** | **B** | **D** | **C** | **R** | **B** | **D** | **C** | **R** | **B** | **D** |
| 20 | 1.4 | 0.7 | 0.2 | 0.0 | 0.1 | 0.3 | 0.0 | -8.4 | 0.1 | 2.1 | 0.0 | 0.0 | 0.2 | 2.2 | 0.0 | -1.2 |
| 30 | 0.6 | 0.4 | 0.2 | 0.0 | 0.0 | 0.1 | 0.0 | -5.9 | 0.1 | 1.0 | 0.1 | 0.0 | 0.1 | 0.9 | 0.1 | -3.8 |
| 40 | 0.4 | 0.3 | 0.1 | 0.0 | 0.0 | 0.1 | 0.0 | -7.5 | 0.1 | 1.2 | 0.1 | 0.0 | 0.2 | 1.2 | 0.1 | -3.5 |
| 50 | 0.2 | 0.1 | 0.2 | -0.2 | 0.0 | 0.1 | 0.1 | -6.5 | 0.0 | 0.8 | 0.3 | -0.9 | 0.1 | 0.9 | 0.3 | -5.7 |
| 70 | 0.2 | 0.1 | 0.0 | -0.2 | 0.0 | 0.0 | 0.0 | -2.3 | 0.0 | 0.5 | 0.1 | -0.3 | 0.1 | 0.5 | 0.1 | -0.4 |
| 100 | 0.1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | -1.0 | 0.0 | 0.3 | 0.0 | 0.0 | 0.0 | 0.3 | 0.0 | -0.1 |




(a) Stream+Web (b) Stream+Web-R

(c) P2P+VoIP (d) P2P+VoIP-R

Fig. 3: $p_{\text{em}}$ development for both runs of the extension procedure and all VN types. Suffix R denotes the second run with additional resources for the substrates.

often a specific $p_{\text{em}}$ occurred depending on the number of added VNs. Stream and Web have been combined into one graph, as have P2P and VoIP, because they show very similar behaviour. Clearly visible is the shift of the clusters to the right, i.e. more VNs can be added before the substrate is full. Figure 3b shows that for stream and web VNs, the added resources cause a substantial fraction of instances to stay at $p_{\text{em}} = 1$ during the first five added VNs. This is in contrast to the situation before additional resources were added, as seen in Figure 3a. After five added VNs, most instances show a $p_{\text{em}}$ smaller than 0.2. A similar, although not as pronounced, development can be seen for P2P and VoIP VNs. It can be seen that during the first run of the extension procedure (Figure 3c) it happens very often that the initial VNMP-DRL instance cannot be extended with an additional VN. Nearly every initial instance has $p_{\text{em}} \leq 0.12$. After adding additional resources, $p_{\text{em}}$ of the initial instance covers the whole range of possibilities. Adding further VNs reduces $p_{\text{em}}$ far more rapidly than for stream and web VNs. On the whole the added resources are more effective for influencing $p_{\text{em}}$ for stream and web VNs than they are for P2P and VoIP VNs. There are two possible explanations for this behaviour. First

TABLE V: Average $p_{\mathrm{em}}$ in percent for the first run of the extension procedure and for the second run with additional substrate resources.

| | Stream | | Web | | P2P | | VoIP | |
|---|---|---|---|---|---|---|---|---|
| **Size** | **1st** | **2nd** | **1st** | **2nd** | **1st** | **2nd** | **1st** | **2nd** |
| 20 | 13.6 | 38.8 | 18.4 | 59.7 | 7.4 | 20.2 | 7.9 | 20.4 |
| 30 | 13.1 | 39.6 | 14.7 | 60.8 | 3.3 | 18.2 | 3.7 | 15.5 |
| 40 | 13.5 | 42.7 | 11.7 | 60.6 | 4.3 | 15.4 | 3.1 | 14.3 |
| 50 | 15.7 | 40.7 | 9.0 | 51.7 | 11.6 | 13.6 | 7.2 | 14.7 |
| 70 | 12.7 | 33.5 | 11.3 | 31.5 | 2.6 | 11.7 | 2.9 | 15.4 |
| 100 | 8.0 | 31.7 | 12.2 | 30.9 | 0.0 | 0.9 | 0.0 | 0.6 |

of all, since the first run of the extension procedure for P2P and VoIP VNs often could not extend the initial instance, the collected failure reasons only contain reasons why adding one VN might fail. Therefore, after adding resources, only adding the first VN works well and $p_{\mathrm{em}}$ rapidly approaches zero afterwards. Another possible explanation is that the used settings for determining where and how much resources are added might not cause enough resources to be added so that multiple VNs can be embedded in the substrate. The chosen settings work well for stream and web VNs, but might be too conservative for P2P and VoIP VNs, which require more resources per VN.

In Table V, we show the average embedding probability for all instances generated by both runs of the extension procedure. It can be seen that the greatest gains have been achieved for web VNs (which also had the largest resource change). A close second are stream VNs, where less than 1% change in resources increased $p_{\mathrm{em}}$ by 20% and more on average. The improvements for P2P and VoIP VNs were not as great, especially for instances of size 100 where the ILP fails to extract useful failure reasons due to time and memory constraints.

## VI. Conclusion

In this work we showed that a simple ILP can be successfully employed to find bottlenecks in a substrate. Our results exposed that less than one percent of additional resources in the substrate network can increase the probability that additional VNs can be embedded by 20% or more. We also showed that different use cases (i.e. different VNs) lead to different resources being added to the substrate, so Virtual Network Operators (VNOs) are able to optimize their networks to cater best to the VN types they encounter the most. Furthermore, we think that it is useful to identify bottlenecks using the same methodology as for the embedding process later on in order to avoid side effects due to remapping of VNs (e.g. caused by reduced path delays).

Thus, a VNO could monitor its current network situation and decide whether the lease of additional resources on a specific location is necessary to provide a certain service availability. Knowing these bottlenecks in advance offer the VNO the possibility to preconfigure these resources in order to reduce the embedding time in the presence of a new VN request.

## References


[1] "Gush," http://gush.cs.williams.edu/. [Online]. Available: http://gush.cs.williams.edu/

[2] "Teagle," http://www.fire-teagle.org/. [Online]. Available: http://www.fire-teagle.org/

[3] "RSpec - ProtoGENI," http://www.protogeni.net/trac/protogeni/wiki/RSpec. [Online]. Available: http://www.protogeni.net/trac/protogeni/wiki/RSpec

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, p. 69–74, 2008.

[5] J. Kelly, W. Araujo, and K. Banerjee, "Rapid service creation using the JUNOS SDK," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, p. 56–60, Jan. 2010. [Online]. Available: http://doi.acm.org/10.1145/1672308.1672320

[6] J. Wroclawski, "The Use of RSVP with IETF Integrated Services," RFC 2210 (Proposed Standard), Internet Engineering Task Force, Sep. 1997. [Online]. Available: http://www.ietf.org/rfc/rfc2210.txt

[7] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC 2474 (Proposed Standard), Internet Engineering Task Force, Dec. 1998, updated by RFCs 3168, 3260. [Online]. Available: http://www.ietf.org/rfc/rfc2474.txt

[8] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Service," RFC 2475 (Informational), Internet Engineering Task Force, Dec. 1998, updated by RFC 3260. [Online]. Available: http://www.ietf.org/rfc/rfc2475.txt

[9] D. Stezenbach, M. Hartmann, and K. Tutschku, "Parameters and challenges for virtual network embedding in the future internet," Maui, Hawaii, Apr. 2012.

[10] J. Inführ and G. R. Raidl, "Introducing the virtual network mapping problem with delay, routing and location constraints," in *Network Optimization: 5th International Conference, INOC 2011*, ser. LNCS, J. Pahl, T. Reiners, and S. Voß, Eds., vol. 6701. Hamburg, Germany: Springer, June 2011, pp. 105–117.

[11] B. Fortz and M. Thorup, "Internet Traffic Engineering by Optimizing OSPF Weights," Tel-Aviv, Israel, 2000, pp. 519–528.

[12] M. Hartmann, D. Hock, M. Menth, and C. Schwartz, "Objective Functions for Optimization of Resilient and Non-Resilient IP Routing," in $7^{th}$ *International Workshop on the Design of Reliable Communication Networks (DRCN)*, Oct. 2009.

[13] S. Shenker, "Fundamental Design Issues for the Future Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176–1188, Sep. 1995.

[14] M. Menth, R. Martin, and J. Charzinski, "Capacity Overprovisioning for Networks with Resilience Requirements," in *ACM SIGCOMM*, Pisa, Italy, Sep. 2006.

[15] D. Hock, M. Menth, M. Hartmann, C. Schwartz, and D. Stezenbach, "ResiLyzer: A Tool for Resilience Analysis in Packet-Switched Communication Networks," in *GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB) and Dependability and Fault Tolerance (DFT)* , Essen, Germany, Mar. 2010.

[16] M. Menth, M. Duelli, R. Martin, and J. Milbrandt, "Resilience Analysis of Packet-Switched Communication Networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, pp. 1950 – 1963, Dec. 2009.

[17] A. Fischer, J. F. Botero, M. Duelli, D. Schlosser, X. Hesselbach, and H. de Meer, "ALEVIN - A Framework to Develop, Compare, and Analyze Virtual Network Embedding Algorithms," *Electronic Communications of the EASST, Kommunikation in Verteilten Systemen 2011*, vol. 37, Mar. 2011.

[18] J. Inführ and G. R. Raidl, "The VNMP-DRL benchmark set." [Online]. Available: https://www.ads.tuwien.ac.at/projects/optFI/

[19] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, ser. SIGCOMM '02. New York, NY, USA: ACM, 2002, pp. 133–145.

[20] R. Govindan and H. Tangmunarunkit, "Heuristics for internet map discovery," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 3, Mar. 2000, pp. 1371 –1380 vol.3.

[21] H. Burch and B. Cheswick, "Mapping the internet," *Computer*, vol. 32, no. 4, pp. 97 –98, 102, Apr. 1999.

[22] IBM ILOG, "CPLEX 12.4." [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer