Electronic Communications of the EASST
Volume 37 (2011)

Workshops der wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen 2011 (WowKiVS 2011)

ALEVIN - A Framework to Develop, Compare, and Analyze Virtual Network Embedding Algorithms

Andreas Fischer, Juan Felipe Botero, Michael Duelli, Daniel Schlosser, Xavier Hesselbach, Hermann de Meer

12 pages

Guest Editors: Horst Hellbrück, Norbert Luttenberger, Volker Turau
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

ALEVIN - A Framework to Develop, Compare, and Analyze Virtual Network Embedding Algorithms

Andreas Fischer[1], Juan Felipe Botero[2], Michael Duelli[3], Daniel Schlosser[3], Xavier Hesselbach[2], Hermann de Meer[1]

[1]andreas.fischer@uni-passau.de, demeer@uni-passau.de
Universität Passau, Innstr. 43, 94032 Passau, Germany
[2]jfbotero@entel.upc.edu, xavierh@entel.upc.edu
Universitat Politècnica de Catalunya, Jordi Girona Street 1-3, 08034 Barcelona, Spain
[3]duelli@informatik.uni-wuerzburg.de, schlosser@informatik.uni-wuerzburg.de
University of Würzburg, Am Hubland, 97074 Würzburg, Germany

Abstract: Network virtualization is recognized as an enabling technology for the Future Internet. Applying virtualization of network resources leads to the problem of mapping virtual resources to physical resources, known as "Virtual Network Embedding" (VNE). Several algorithms attempting to solve this problem have been discussed in the literature, so far. However, comparison of VNE algorithms is hard, as each algorithm focuses on different criteria. To that end, we introduce a framework to compare different algorithms according to a set of metrics, which allow to evaluate the algorithms and compute their results on a given scenario for arbitrary parameters.

Keywords: Virtual network, network virtualization, network embedding, embedding algorithms, evaluation framework

1 Introduction

Virtualization of network resources has been identified as key technology for Future Internet research [BFM10] and is actively used in current research testbeds [CJ09, SGH+10]. By virtualizing both node and link resources of a substrate network, multiple virtual network topologies with widely varying characteristics can be created and cohosted on the same physical hardware. Moreover, the abstraction introduced by the resource virtualization mechanisms allows network operators to manage and modify networks in a highly flexible and dynamic way. A Virtual Network (VN), in this sense, is a combination of active and passive network elements (network nodes and network links) on top of a Substrate Network (SN). Virtual nodes are interconnected through virtual links, forming a network that can be represented by a directed graph, where the virtual nodes correspond to the nodes in the graph and the virtual links correspond to the edges.

The flexibility gained through such an approach can be used to increase sustainability (in terms of cost-effectiveness) of deployed network hardware. By dynamically mapping virtual resources onto physical hardware, the benefit gained from existing hardware can be maximized. However, this leads to an optimization problem: In order to achieve efficient operation of such virtualized networks, it is paramount to develop algorithms that distribute virtual resources on

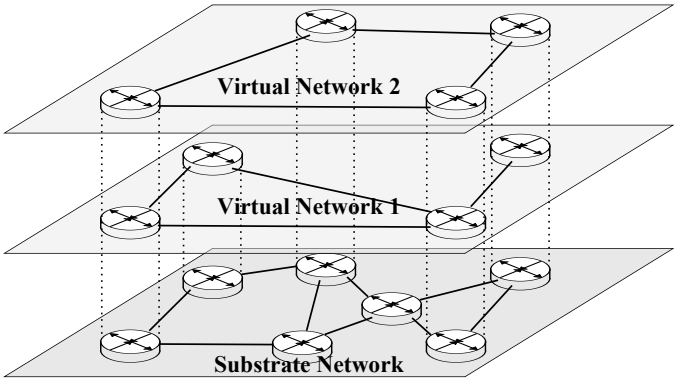

Figure 1: Two Virtual Networks embedded in a Substrate Network.

a physical infrastructure in an optimal way. This optimality can be computed with regard to different parameters, ranging from load distribution over energy-efficiency to security of the networks. The problem of embedding virtual networks in a substrate network in an optimal way is known as the Virtual Network Embedding (VNE) problem. Several algorithms have already been proposed to solve this problem. Comparison of these algorithms is difficult, as typically each algorithm optimizes only for a subset of all possible network parameters. In order to compare the efficiency of these algorithms, a set of appropriate metrics has to be found. Here, a framework is presented that allows to compare VNE algorithms according to different metrics.

The remainder of this paper is organized as follows: Section 2 discusses the VNE problem in detail. Section 3 gives an overview of the different parameters that can be taken into account for the embedding. Section 4 presents VNE algorithms that have been proposed so far. Section 5 presents the framework and metrics proposed in this paper to compare different VNE algorithms. Finally, Section 6 concludes this paper and discusses our intended future work in this area.

2 The Virtual Network Embedding Problem

2.1 Overview

The Virtual Network Embedding problem deals with the efficient mapping of a set of Virtual Network Requests (VNRs) to substrate nodes and links. A VNR is a set of virtual nodes that must be mapped to a set of substrate nodes with sufficient resources to accomplish the requirements, and a set of virtual links to be mapped to a set of paths in the substrate network. This embedding should optimize the allocation of physical resources. Embeddings can be optimized with regard to performance (e.g. CPU capacity, link bandwidth), energy-efficiency (e.g. power usage of a node), security (e.g. node reliability, link encryption), or other parameters. Moreover, several side constraints, such as fixed mappings for certain virtual nodes, might further influence the embedding.

Figure 1 shows two VNs that are embedded in a SN. A node in the SN can host several virtual nodes. Mapping of virtual links is more complex in this example. A virtual link is mapped onto a path in the SN (i.e. several physical resources are combined to form a virtual resource).

Table 1: Terminology used throughout this paper

Term	Description
$G = (V,A)$	G is a graph representing a SN. It consists of a set of vertices (V) and a set of directed edges (arcs, A) connecting the vertices.
$G^k = (V^k,A^k)$	G^k is the graph of the k-th Virtual Network request. Like G, it consists of a set of vertices (V^k) connected with a set of arcs (A^k).
$f : V^k \to V$	f is the function that maps virtual nodes to nodes in the substrate network.
$P = \{G' \mid G' \subset G\}$	P is the set of all directed paths G' in G
$g : A^k \to 2^P \setminus \emptyset$	g is the function that maps virtual links to a set of directed paths (an element of the power set of all paths 2^P) in the substrate network.
$HH : A^k \times V \to \mathbb{R}$	HH is a function that defines the demand of a virtual link $(i^k, j^k) \in A^k$ on a given Hidden Hop $l \in V$.

Mapping a virtual link to a path in the SN obviously uses resources of the physical links on the path. However, there are also physical nodes on the path that will be traversed by the virtual link. These are called "Hidden Hops" here. The virtual link will also consume resources of all Hidden Hops on the paths. Furthermore, several virtual links can use the same physical link (i.e. a physical resource can be partitioned into several virtual resources). For the mapping of virtual nodes and virtual links, several algorithms can be considered. Due to the differences in node and link mapping, most algorithms treat node and link mapping as separate steps. It can be shown that the resulting network embedding problem is *NP*-hard. Practical algorithms, therefore, use heuristic approaches to obtain a near optimum value.

Algorithms solving the VNE problem come in two forms: offline algorithms and online algorithms. Offline algorithms take a given set of VNRs together with the description of a SN and compute a near optimal embedding for these requests. While this approach achieves good results with regard to optimality, it does not consider a dynamic arrival process of the VNRs (i.e. each VNR arrives at a different time and must be mapped in real time). Online algorithms, on the other hand, take VNRs on a FIFO-basis, redistributing virtual resources as the requests arrive. While this approach is better suited to deal with high dynamicity, it tends to come at the cost of less optimal solutions. Things might be further complicated by looking at heterogeneous resources in the SN (i.e. resources that are described by different sets of parameters or that have widely varying attributes). Adapting algorithms to support such an assumption therefore remains an open issue for now.

2.2 Formal problem description

Formally, the VNE problem can be described with the terminology given in Table 1. A physical network is given as a directed graph $G = (V,A)$ where vertices represent the nodes in the network and directed edges represent the links between nodes. On this network, virtual networks – each described by its own directed graph $G^k = (V^k,A^k)$ – are mapped by assigning a physical node for each virtual node and a physical path for each virtual link.

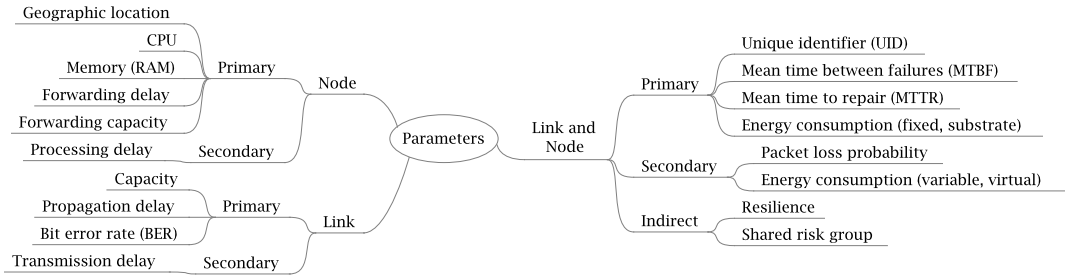

Figure 2: A categorization of parameters that can be considered in the VNE problem.

The function performing this operation is the mapping function that has to be realized by the VNE algorithms. This function can be divided into node and link mapping. The node mapping function $f : V^k \to V$ maps nodes from the virtual network requests to substrate nodes. Likewise, the link mapping function is defined as $g : A^k \to 2^P$ where 2^P consists of all sets of directed paths in the SN. If g maps to a set with more than one element, this allows to consider algorithms solving the VNE problem with the use of multipath routing. Otherwise, the element in the set is the path used to realize the virtual link. Both functions may not exceed the resources of either a node or a link in the SN. An optimal embedding then is a function that satisfies all of the above restrictions and additionally satisfies a given optimization objective (e.g. to maximize the spare resources in a SN).

3 Parameters

In this section, we present a categorization of parameters for substrate as well as virtual nodes and links. In addition, a mapping of parameters is given that affect hops on a path through the SN which are hidden on a virtual link.

3.1 Categorization

We divide the parameters for nodes and links in three categories regarding their mutability, accessibility, and interdependency with other parameters:

- *Primary parameters* are fixed properties of the node or link itself. These parameters – e.g. CPU or memory of a node, capacity and delay of a link – are capabilities that do not depend on another node's or link's state, but only on its utilization. Primary parameters can be directly requested by a virtual network. We denote the set of primary parameters by *PRI*.
- *Secondary parameters* are derived attributes of a node or link. These parameters depend on the state of its node or link, i.e. other primary and/or secondary parameters. For instance, the loss probability of a node depends on its CPU load, which can, in turn, indicate the queue size at a router, and the transmission delay of a virtual link is the sum of the transmission and propagation delay of all traversed substrate links and the processing and forward delay of all traversed substrate nodes.

To prevent cyclic dependencies, we only consider the dependency on primary parameters. Secondary parameters can also be directly requested by a virtual network. We denote the set of secondary parameters by *SEC*.

- *Indirect parameters* are requested for a whole substrate path including traversed links and nodes. For instance, the demand for a resilient virtual link comprises that traversed substrate nodes and links fulfill certain failure probabilities and that the primary path is disjoint from the backup path in the substrate to avoid a *shared risk group*.
  These parameters are not specific to nodes and links. Hence, indirect parameters are not proportional to the number of virtual networks embedded. Indirect parameters can only be *indirectly* requested by a VNR.

In Figure 2, we illustrated some common parameters that can be considered in the VNE problem.

It is important to note, that some of the parameters have to be handled differently for substrate and virtual network entities. For instance, CPU resource is a primary (i.e. fixed) parameter for a substrate node while it is a secondary parameter on a virtual link which comprises the CPU resource on the nodes that are traversed in the substrate to realize this virtual link. As a consequence, we have to define a mapping of virtual to substrate parameters including requests to *hidden hops*.

3.2 Parameter Mapping

To declare secondary parameters, we need to define how the mapping of primary parameters to secondary parameters is calculated. Furthermore, we need a mapping how traffic affects a *hidden hop*.

Primary to Secondary

We define *PRI* and *SEC* as the set of primary and secondary parameters respectively. Secondary parameters are calculated from a set of surjective functions $map_{sec} : PRI^n \rightarrow SEC, sec \in SEC, 1 \leq n \leq |PRI|$. Once all primary substrate node and link parameters have been defined, a planning algorithm would compute each secondary parameter $sec \in SEC$, based on the node characteristics and using the mapping function map_{sec}.

Hidden Hops Mapping

Hidden hops, introduced in [BHFD13], makes reference to the intermediate nodes of a directed path in the SN that is mapping a specific virtual link of a VNR. We claim that a hidden hop entails a resource demand because it has to perform packet forwarding of the traffic that will pass through this virtual link.

Hidden-hop demand depends on the demand in the virtual links and the node type. Therefore, for each hidden-hop l in a SN path mapping a virtual link, the demand is a function $HH(i^k, j^k, l), l \in V, (i^k, j^k) \in A^k$ calculated from a combination of the virtual link demand and the node characteristics. It closely depends on the specific mapping scenario and equipment type.

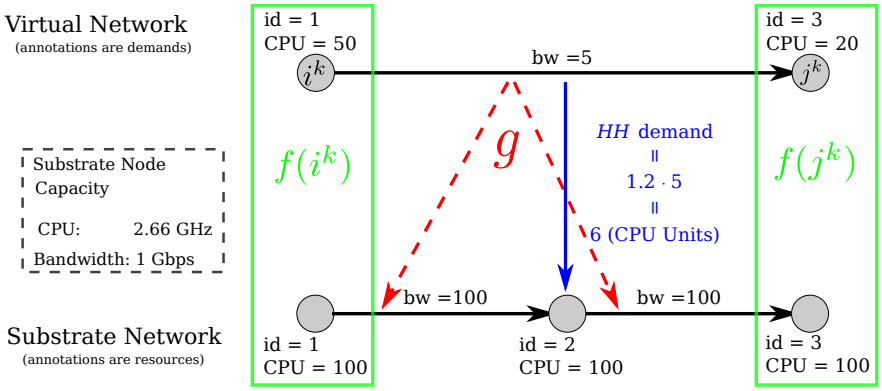

Figure 3: Illustration of parameter categories and their application.

As an example of a hidden hop demand calculation, let's consider a SN transporting packets with size of 1500 bytes among substrate nodes as illustrated in Figure 3 (node type extracted from [LYG10]). Assuming that CPU and bandwidth (BW) capacity units are scaled in a way that 100 CPU units = 2.66 GHz and 100 BW units = 1Gbps, the hidden hops CPU demand, due to the virtual link bandwidth demand, is calculated based on the number of cycles used to process a packet in the node when it is totally loaded (40000, in this case). One bandwidth demand unit is equivalent to $\sim$ 833 packets/sec, therefore, the amount of CPU units needed to process it is 1.2. That is, the hidden hops belonging to the SN path mapped to the virtual link (i^k, j^k) CPU demand will be 1.2 multiplied by the units of BW demand of the virtual link (i^k, j^k).

4 Main Virtual Network Embedding Algorithms

Due to their novelty and relevance, we selected a set of algorithms to be implemented in the first version of this framework. These algorithms only consider mapping of VNs - protocols for changing VNs are not considered here. We propose a taxonomy to classify the different parameters associated with them (Table 2). Next, the procedures used by each VNE algorithm are described.

4.1 Stress Based Approach

One of the first offline approaches to solve VNE is presented in [ZA06] where the objective is to maintain a balanced link and node *stress* in the SN. Stress of a SN node is the number of virtual nodes assigned to it whereas stress of a SN link is the number of virtual links containing it. To achieve that objective, the combination of both, maximum node stress and maximum link stress, is minimized. A heuristic approach is proposed that divides a VNR into a number of connected sub-VNs, each with star topology. After VN subdivision, virtual node and link mappings are realized in each sub-VN. First, node mapping is solved using a greedy algorithm and then, link mapping is solved using a shortest path solution between the mapped nodes.

Table 2: VNE algorithms taxonomy

Classification Parameter		Description
Environment Type	Online	VNRs arrive dynamically and they are not known in advance
	Offline	All VNRs are known in advance so the VN requests service order can be organized in advance
Algorithm Type	Dynamic	After several VNEs are performed the algorithm proposes mechanisms to reoptimize performed embeddings
	Static	No reoptimizing mechanisms are performed
Virtual Link Assignment	Multi-Path (MP)	Multiple SN paths with different splitting ratio to map each virtual link
	Single-Path (SP)	Each virtual link is mapped to just one SN path.
Hidden Hops	Considered	Hidden Hop demands are considered when VNE is performed
	Not Considered	No Hidden Hop demand is considered.

4.2 Path Splitting VNE Approach

The maximization of a long average *revenue*, i.e. the weighted sum of VNR's bandwidth and CPU demands, is the objective of the algorithm proposed in [YYRC08]. The online algorithm queues a group of incoming VNRs (decreasingly ordered by revenue) during a time window and tries to efficiently allocate them. If one request is not accepted it is sent to the request queue waiting for resources release in the SN until a specified timeout, then it is dropped of the queue.

Virtual node and link mappings for each VNR are performed separately. Node mapping is performed using a greedy algorithm that maps virtual nodes with higher revenues to SN nodes with higher "available resources". Link mapping is accomplished in two ways: One is the multi-path (MP) optimal solution that avoids the *NP*-completeness of the problem and the second is to map each virtual link to the shortest-path, accomplishing capacity constraints, between the corresponding mapped nodes in the SN.

4.3 Coordinated Node and Link Mapping VNE Approach

An online algorithm proposing a new virtual node mapping stage is presented in [CRB09]: Virtual link mapping is performed as in [YYRC08]. A new set of important node constraints are added in the node mapping; geographical location for substrate and virtual nodes and a non-negative distance per VNR indicating how far a node of the VNR can be of its demanded location.

The node mapping stage is performed by defining an augmented graph over the substrate network; introducing a set of meta-nodes, one per virtual node, each connected to a cluster of candidate SN nodes obeying location and capacity constraints. The algorithm solves the VNE problem by using a Mixed Integer Programming (MIP). Its objective is to minimize the *cost*, i.e. the weighted sum of the bandwidth and CPU allocated in the SN links to fulfill VNR demands, of embedding a VNR. To avoid the *NP*-completeness of the MIP, its linear programming relaxation is solved, and the obtained solution is rounded in two ways: deterministically or randomly.

4.4 Subgraph Isomorphism Detection VNE Approach

Lischka and Karl [LK09] have proposed an approach based on the Subgraph Isomorphism Detection problem (SID). In graph theory, an isomorphism of graphs G and H ($G \simeq H$) is a bijection between the vertex sets of G and H, $m : V(G) \rightarrow V(H)$, such that any two vertices i and j of G are adjacent in G if and only if $m(i)$ and $m(j)$ are adjacent in H. The NP-complete SID problem tries to find a subgraph G_{sg} of G ($G_{sg} \subset G$) such that $G_{sg} \simeq H$.

The VNE problem can be reduced to this well known problem. They propose an heuristic algorithm consisting on finding an isomorphic subgraph (representing the VNR), accomplishing the demands, inside the substrate network. The main contribution of this approach is to realize in the same stage the node and link mapping This feature helps saving run-time cost when bad link mapping decisions are taken.

4.5 Topology-Aware and Re-Optimization VNE Approach

Two important contributions to improve the acceptance rate of VNRs are made in [BCB10]:

Topology-Aware Embedding: There are critical SN links and nodes contributing in the network *partition* when mapped; a link or node is susceptible of *partition* when its removal divides the SN in two different networks. These critical resources must be considered in the objective function of current approaches.

Re-optimizing Bottlenecked Embeddings: As VNs are embedded with an associated lifetime, arbitrary values of them can cause the whole SN embedding to become inefficient. Periodic reconfiguration schemes do not work for real situations (incurred overhead of virtual links and nodes reallocation). The approach proposed here is to act whenever a VNR gets rejected. The solution consists on identifying virtual links and nodes causing VNR rejection and then, relocate them to less critical regions of the SN.

4.6 Hidden Hops VNE Approach

In previous work [BHFD13], the important and realistic concept of hidden hops was introduced in VNE. This work considered the offline version of the algorithm. The virtual node mapping is not considered, whereas virtual link mapping is performed following k-shortest path approach, taking into account the hidden hops demand when each virtual link is mapped.

This work proposes to realize the VNE with demands of CPU in nodes and bandwidth in nodes but also considers VNR without explicit demand requirements. The proposed algorithm maps the requests with demands, trying to maximize the spare resources after embedding of each VNR (equivalent to minimize the cost). Then, remaining VNRs are mapped, by distributing equal resources to each.

4.7 Summary

To summarize this section, we built Table 3 that lists the characteristics of the presented VNE algorithms. It can be noted that only CPU in nodes and bandwidth in links are considered by the main algorithms to solve VNE. The introduction of a more complete set of parameters to the main algorithms (possibly adding more constraints) could be helpful in the consideration of

Table 3: Characteristics of the main proposals to solve VNE.

Reference	Optimization Objective					Link and Node Constraints		Taxonomy				Link Heuristic	Node Heuristic
	Balanced Stress	Revenue	Cost	Feasible Embedding	Cost + CI	Unconstrained	CPU + Bandwidth	Environment Type	Algorithm Type	Virtual Link Assignment	Hidden Hops		
[ZA06]	**X**					**X**		Offline	STA/DYN	SP	NC	SH-P	GS
[BHFD13]			**X**				**X**	Offline	STA	SP	C	SH-P	NC
[YYRC08]		**X**					**X**	Online	STA	MP	NC	MFP	GAR
[CRB09]			**X**				**X**	Online	STA	MP	NC	MFP	R-MIP
[LK09]				**X**			**X**	Online	STA	SP	NC	SID	SID
[BCB10]					**X**		**X**	Online	DYN	MP	NC	MFP	R-MIP

STA:Static SP: Single-Path NC: Not Considered SH-P: Shortest Path GS: Greedy Stress
DYN:Dynamic MP: Multi-Path C: Considered SID: Subgraph Isomorphism Detection
GAR: Greedy Available Resources R-MIP: Rounding Mixed Integer Program MFP: Multicommodity Flow Problem

different embedding goals (Energy efficiency, resiliency, QoS related parameters, etc.). It would also be very useful to evaluate an algorithm in both, offline and online environments; this would allow to compare different kind of algorithms in more detail. ALEVIN will consider a larger and extendable set of network parameters (see Section 3). It will also allow to evaluate algorithms in online and offline environments.

5 Framework and Evaluation Metrics

We developed a software framework to support the development of new algorithms, ease their comparison and analysis, and apply arbitrary evaluation metrics. In this section, we introduce this framework – named *ALEVIN* – and outline its capabilities regarding parameters, algorithms, and evaluations. After implementing new algorithms and defining their optimization parameters, this framework can be used to test and compare those algorithms.

5.1 ALEVIN – ALgorithms for Embedding VIrtual Networks

The focus in the development of ALEVIN [VNR10] is on modularity and efficient handling of arbitrary parameters for resources and demands, as described in Section 3, and the support of integration of new and existing algorithms (cf. Section 4) and evaluation metrics. For platform independency, ALEVIN is written in Java. ALEVIN's *graphical user interface* (GUI) and multi-layer visualization component is based on the MuLaViTo project [DO10] which enables us to visualize and handle the substrate and arbitrary virtual networks as directed graphs. Figure 4 depicts the architecture of ALEVIN and highlights the modular interaction of parameters for

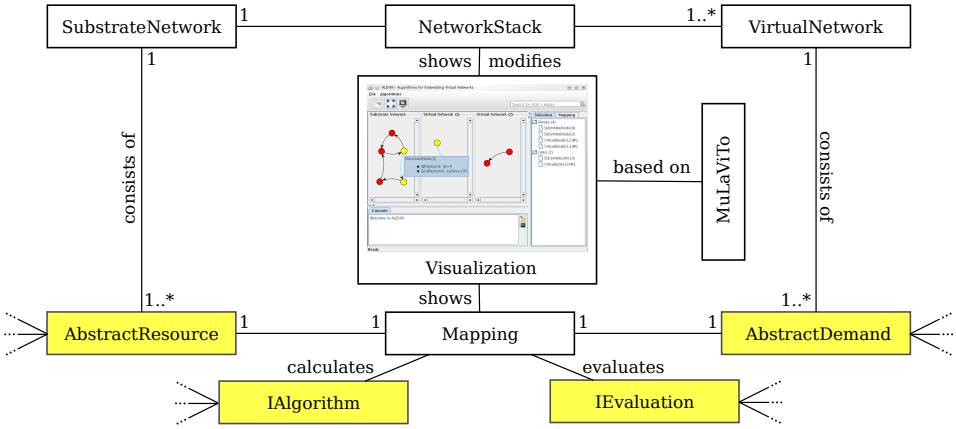

Figure 4: The framework ALEVIN and its modularity in parameters, algorithms, and evaluations.

substrate as well as virtual networks, algorithms, and evaluation.

ALEVIN provides the ability to illustrate the deployment of resources in the SN and demands in an arbitrary number of VNs as well as the mapping of demands on resources calculated by a VNE algorithm. Moreover, ALEVIN can be used to create VNE scenarios as well as import and export them using an XML-based exchange format. ALEVIN is completely modular regarding the addition of new parameters to the VNE model. Using the *visitor design pattern* in a sophisticated way, we are able to avoid any casts to concrete demand/resource classes. Thus, the number of parameters is not performance-relevant and a convenient implementation of arbitrary parameters is possible. To increase ALEVIN's modularity and to make it a flexible and extensible platform to compare existing and upcoming algorithms, the implementation of algorithms is kept independent of the resource/demand implementation. To that end, a simple interface is provided defining the rough structure of an algorithm and connecting its output to the GUI as illustrated in Figure 4.

5.2 Evaluation and Comparison of VNE Algorithms

ALEVIN also is our basis for the evaluation and comparison of VNE algorithms. Therefore, ALEVIN provides a simple interface to add evaluation metrics which are independent of the implemented parameters and algorithms. This further emphasizes the modularity of ALEVIN.

During and after the calculation of a mapping of either one or several virtual network demands, different evaluation metrics can be applied. Table 4 list several evaluation metrics. These evaluation metrics can be combined in an optimization objective/strategy for a whole VNE scenario. For instance, a optimization objective for a VNE scenario might be to minimize the maximum link utilization, maximize the residual bandwidth capacity, minimize the total cost for the substrate provider, maximize the acceptance ratio, minimize the overall energy consumption, or minimize run time. Due to the ability to combine arbitrary evaluation metrics and optimization objectives in combination with arbitrary parameters and a simple interface for VNE algorithms, ALEVIN is a powerful framework for the comparison and analysis of VNE algorithms regarding different evaluation metrics and optimization objectives.

Table 4: Evaluation metrics used by the algorithms in Section 4.

Metric	Description
Node stress	The number of virtual nodes mapped on top of a substrate node
Link stress	The number of virtual links using a substrate link
Revenue	A weighted sum of the demands (CPU, bandwidth, ...) of a virtual network
Cost	A weighted sum of the resources allocated by a virtual network mapping
Acceptance ratio	Ratio of number of accepted to total virtual network request
Node utilization	Utilization measured in each resource-type node parameter
Link utilization	Utilization measured in each resource-type link parameter
Cost/revenue ratio	–
Splitting ratio	Percentage of virtual network requests using path splitting

6 Conclusion and Future Work

With the expected rise of network virtualization as enabler for flexible and autonomic network management, the decision on how to map virtual resources to physical resources is gaining importance. Achieving optimal VNEs, therefore, is an important problem to solve for Future Internet infrastructures. In this paper, the VNE problem has been described in detail. Several algorithms solving the VNE problem have been presented. It was argued that it is necessary to compare these algorithms according to different metrics. This will allow to rank different algorithms against each other and choose the best solution for a given embedding problem. ALEVIN - a framework that allows to implement these algorithms and compute solutions for different networks - was presented and discussed in this paper. ALEVIN is developed as *open source* and will be released under *GNU General Public License* (GPL) and *GNU Lesser General Public License* (LGPL). ALEVIN as well as the VNE exchange format will be made publicly available on [VNR10].

Building on this work, we plan to extend the implementation of the software. More algorithms should be added and rated. Moreover, the investigation of these algorithms leads to interesting new problems: both, energy-efficiency and security related problems in VNE have not been investigated in depth, yet. A promising way forward would be to investigate how existing algorithms can be adapted to support these additional criteria.

Acknowledgements: The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013] [FP7/2007-2011]) in the context of the "Euro-NF" Network of Excellence (grant agreement no. 216366) through the Specific Joint Developments and Experiments Project "Virtual Network Resource Embedding Algorithms" (VNREAL) and in the context of the ResumeNet project, grant agreement no. 224619. It has been also partially supported by the "Comissionat per a Universitats i Recerca del DIUE" from the "Generalitat de Catalunya", the Social European Budget ("Fons Social Europeu"), and by the Federal Ministry of Education and Research of the Federal Republic of Germany (BMBF Förderkennzeichen 01BP0775) in the context of the EUREKA project "100 Gbit/s Carrier-Grade Ethernet Transport Technologies (CELTIC CP4-001)".

Bibliography

[BCB10] N. F. Butt, N. M. Chowdhury, R. Boutaba. Topology-Awareness and Reoptimization Mechanism for Virtual Network Embedding. In *Networking*. Pp. 27–39. 2010.

[BFM10] A. Berl, A. Fischer, H. de Meer. Virtualisierung im Future Internet - Virtualisierungsmethoden und Anwendungen. *Informatik-Spektrum* 33(2):186–194, Apr. 2010.

[BHFD13] J. F. Botero, X. Hesselbach, A. Fischer, H. De Meer. Optimal Mapping of Virtual Networks with Hidden Hops. *Telecommunication Systems: Modelling, Analysis, Design and Management "Accepted - To be published"*, 2013.

[CJ09] J. Carapinha, J. Jiménez. Network virtualization: a view from the bottom. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*. VISA '09, pp. 73–80. ACM, New York, NY, USA, 2009.

[CRB09] N. M. M. K. Chowdhury, M. R. Rahman, R. Boutaba. Virtual Network Embedding with Coordinated Node and Link Mapping. In *Proc. IEEE INFOCOM*. Apr. 2009.

[DO10] M. Duelli, J. Ott. MuLaViTo – Multi-Layer Visualization Tool. Nov. 2010. http://mulavito.sf.net

[LK09] J. Lischka, H. Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*. Pp. 81–88. New York, USA, Aug. 2009.

[LYG10] Y. Liao, D. Yin, L. Gao. Europa: efficient user mode packet forwarding in network virtualization. In *INM/WREN'10: Proceedings of the 2010 internet network management conference on Research on enterprise networking*. Pp. 6–6. USENIX Association, Berkeley, CA, USA, 2010.

[SGH+10] D. Schwerdel, D. Günther, R. Henjes, B. Reuther, P. Müller. German-Lab Experimental Facility. *Future Internet Symposium (FIS) 2010*, 9 2010.

[VNR10] VNREAL. ALEVIN – ALgorithms for Embedding VIrtual Networks. Nov. 2010. http://alevin.sf.net

[YYRC08] M. Yu, Y. Yi, J. Rexford, M. Chiang. Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration. *ACM SIGCOMM CCR* 38(2):17–29, Apr. 2008.

[ZA06] Y. Zhu, M. Ammar. Algorithms for assigning substrate network resources to virtual network components. In *Proc. IEEE INFOCOM*. Pp. 2812–2823. Apr. 2006.