Alkmim *et al. Journal of Internet Services and Applications* 2013, **4**:3
http://www.jisajournal.com/content/4/1/3

Journal of Internet Services and Applications
a SpringerOpen Journal

RESEARCH Open Access

Mapping virtual networks onto substrate networks

Gustavo P Alkmim[1*], Daniel M Batista[2] and Nelson LS da Fonseca[1]

Abstract

Network virtualization is a promising technique for building the Internet of the future since it enables the low cost introduction of new features into network elements. An open issue in such virtualization is how to effect an efficient mapping of virtual network elements onto those of the existing physical network, also called the substrate network. Mapping is an NP-hard problem and existing solutions ignore various real network characteristics in order to solve the problem in a reasonable time frame. This paper introduces new algorithms to solve this problem based on 0–1 integer linear programming, algorithms based on a whole new set of network parameters not taken into account by previous proposals. Approximative algorithms proposed here allow the mapping of virtual networks on large network substrates. Simulation experiments give evidence of the efficiency of the proposed algorithms.

Keywords: Virtual networks, Mapping, Future internet

1 Introduction

The minimalism approach of the architecture of the Internet specific network has enabled its global spread. One consequence of this simplicity, known as the ossification of the Internet, has been the impossibility to provide missing features in the original design. These limitations has prevented the development of many possible applications and services, although various attempts have been made to provide some of the features missing in its design [1].

These attempts to overcome the original limitations include various new mechanisms proposed to promote the evolution of the Internet [2,3]. Those based on network virtualization allow the definition of virtual networks composed of virtual routers and links; these are then hosted by routers and links of the real network called "substrate network". Network virtualization permits the coexistence of various protocol stacks and architectures on a single substrate, without the need to modify the actual physical network. Moreover, this approach imposes no restrictions on the protocols and architectures involved.

One of the main issues in network virtualization is the efficient mapping of virtual networks onto the substrate network [4,5]. This mapping determines the allocation of routers and links of the virtual network onto the routers and links of the substrate network. However, the search for the optimal mapping of virtual networks is an NP-hard problem [6].

Various solutions have been proposed for this problem [1,4,5,7]. However, most of them assume certain restrictions to make the problem tractable, such as the consideration that requests for virtual network establishment be previously known [1,7] or that the substrate capacity be infinite [1,8] and on network topology restricted [7].

This paper proposes a novel solution to the mapping problem to helps to overcome such limitations which imposes fewer restrictions than previous proposals. Our proposal does not consider previous knowledge of virtual network requests; but rather considers that the substrate has a finite capacity, although no specific network topology is assumed. It considers more realistic scenarios and, hence, can handle a large number of parameters that impact on the complexity of a solution.

The proposed algorithms require the presence of repositories of software images containing the software and protocols required by virtual networks. These images are used to instantiate the virtual routers on real routers, as illustrated in Figure 1. Since all images must be transferred from the repository to the real router prior to the

*Correspondence: alkmim@lrc.ic.unicamp.br
[1]State University of Campinas, Campinas, Brazil
Full list of author information is available at the end of the article

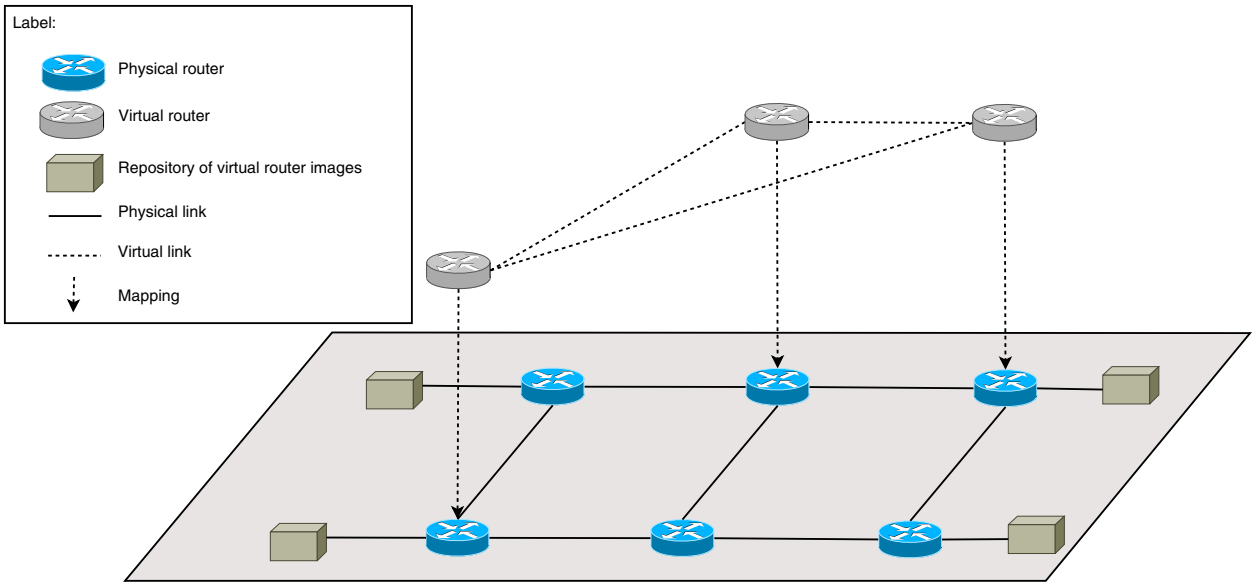

Figure 1 Network architecture: virtual network and real network with repositories of images.

operation of the virtual network, an adequate mapping algorithm must select the image and the path the image transfer should take.

The algorithms proposed here are based on integer linear programming (ILP) formulations designed to minimize the total amount of bandwidth allocated to each virtual network. One of the algorithms is slow but provides optimal solutions, while the others are designed to decrease the run time. Relaxation Techniques for ILP formulations are employed by these approximative algorithms.

The proposed algorithms are efficient since they can provide solutions in a reasonable time frame. The results show that their execution time is acceptable for various scenarios with different virtual networks requirements. The approximative algorithms also produce a reasonable probability of blocking requests in the establishment of virtual network. The algorithms introduced here differ from those in our preliminary investigation [9] in that a two step formulation has been adopted which reduces memory demands thus allowing solutions involving large network substrates with as much as 400 routers.

The paper is organized as follows: Section "Motivation" illustrates the need for a more detailed modelling of the problem. Section "Related work" summarizes related work. Section "Proposed algorithms" introduces the six proposed algorithms. Section "Performance evaluation" presents the performance evaluation of the algorithms and Section "Conclusions and future work" presents the conclusions and suggestions for future work.

2 Motivation

The formulation proposed here models various characteristics of existing operational networks. One of the most important is link delays, which impact on the time needed to instantiate a virtual network, that is, a requirement of service providers. Another important issue is the characteristics of the physical routers.

In general, the algorithms presented in the literature [4] attempt to minimize the amount of resources allocated to requests from virtual networks, but fail to consider the need for transferring image files prior to the instantiation of virtual routers. The following example illustrates the importance of considering link delays and the time for transferring images from the image repository to the physical routers. Figure 2 shows a substrate network with routers, identified as **R1** to **R6**; each router has a different number of processing elements (cores). The available bandwidths of links **E1** to **E5** are labelled in the figure. A repository of images is connected to the router **R4** by the link **E6**. This repository stores the image file **I1** which size is 12.5MB. Each virtual router in the virtual network shown in Figure 3 has two cores and uses the same image **I1**. Moreover, the virtual network must be instantiated in at most 100 seconds.

The router **R1** has no resources available for the allocation of a virtual router since it has a single core and a virtual router requires two cores. Thus, if the transfer of images is ignored, the virtual network using routers (**R2**,**R5**) and link **E4** or routers (**R2**,**R6**) and link **E5** would be instantiated. As a result of such mapping the required image would be transferred to the physical routers via the

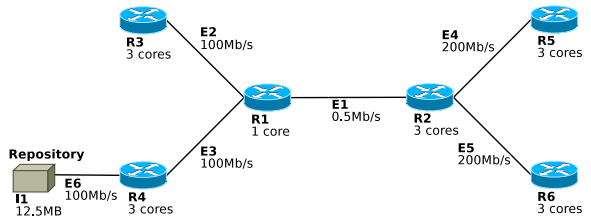

Figure 2 Example of a substrate network.

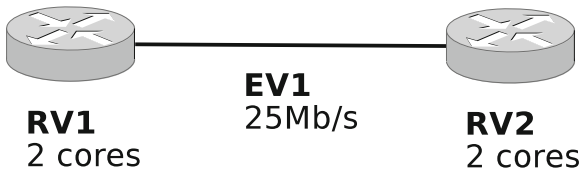

Figure 3 Example of a virtual network.

link **E1**, which has an available bandwidth of only 0.5Mb/s. Thus requiring 404.5 seconds for the transfer, four times as long as the time limit to instantiate the virtual network. Even the use of multicast routing would only reduce this to 202.5 seconds, i.e. twice the limit.

However, the use of the algorithms proposed in this paper would lead to the use of routers (**R3**, **R4**) and links (**E2**, **E3**), since the approach introduced here considers the transfer delay of images and the image transfer would take only four seconds, i.e. much less than the time permitted.

3 Related work

This section summarizes major existing proposals for network virtualization.

The "Cabo" solution [3] is composed of two layers managed by separate providers with infrastructure providers responsible for controlling the elements in the physical layer, and service providers for the provision of network services in the application layer. The approach presented here, however, considers the existence of an additional layer managed by the connectivity providers.

This three-layer architecture is based on the Cabernet architecture [10], which was designed to eliminate some of the limitations in the deployment of virtual services in Wide Area Networks (WAN). This elimination results from by making the infrastructure transparent to the services provider.

The initial design of the Underlay Fused with Overlays (UFO) architecture is first presented in [11]. This solution is also limited to two layers, with the underlay notifying the overlay about changes in network resources. The overlay receives notifications and, in order to increase efficiency and scalability of the virtual networks, can propose routing changes in the underlay. The mapping algorithms in the presented paper can be used in conjunction with the UFO architecture.

In [12], the algorithm Assign was introduced for solving the network testbed mapping problem. This algorithm assumes that a substrate node can only be used by a single request which can lead to under-utilization of the cores of the substrate nodes. Such assumption is not used by the formulations introduced in the present paper, allowing an optimized use of resources. Moreover, the present paper considers the transfer of software images which is neither considered in [12] nor by the Application Component Placement problem [13].

In most of the existing proposals [4-7], the resources considered are limited to bandwidth and routers processing capacity. Some papers [14] do suggest the inclusion of other characteristics as a topic for future work; nonetheless, no solution has yet been published. Our proposal has been able to make several realistic assumptions about resource availability, memory available, the number of processing elements of routers, and the time required to instantiate a virtual router. Our work differ by that in [15] by the modelling of repositories of images in the substrate.

A multi-commodity flow approach was adopted in [16] to maximize the number of virtual networks that could be accommodated on a single substrate network. The substrate has access nodes, which serve as sources and destinations for all traffic, and the core nodes are responsible for the routing of packets. A request for virtual network establishment consists of a list of access nodes of the substrate, as well as of a traffic matrix that represents the amount of traffic transferred between the access nodes listed. Although network capacity is considered, the processing capacity of the nodes is ignored. Furthermore, this approach assumes that the demands of a request are small compared to the available capacity of the network. Unlike [16], algorithms proposed here take into consideration various other characteristics, and no restrictions are imposed on the demands of virtual networks. Another difference is that the algorithm in [16] considers only the edge nodes of the virtual network. The algorithm in [7] is similar to that in [16], except that it is uses a mixed integer quadratic problems branch and bound approach [17] to map the requests.

The network in [5] supports path splitting and path migration, thus allowing the solution to be found in polynomial time. In path splitting, a single virtual link can be mapped onto more than one physical path in the substrate whereas Path migration allows a virtual link to be remapped offline to adapt a solution in the face of changes in resource availability. Although the algorithm in [5] runs in a very short time, it does not consider many of the realistic parameters involved e.g, software images, link delay and the size of images.

In several studies [5,6], the mapping of virtual links is separated from the mapping of virtual routers. Two algorithms were proposed in [4] to integrate the steps, called Deterministic Embedding VN (D-Vine) and Randomized Embedding VN (R-Vine). In the algorithms presented in this paper, however, routers and links can be mapped simultaneously.

In [6], a distributed algorithm to map virtual networks was introduced to balance the load among all routers in the substrate. The experiments presented in [6] show that such as distributed algorithm can generate a large number of control messages which can cause long delays and high overhead for network operation. The algorithms presented here do not overload the network with such messages.

There are various aspects that make the solution of the problem of mapping virtual networks very challenging [5]. The first is the large number of router characteristics. The second is that given resources limitations, there is a need for admission control. The third is the fact that requests for virtual network establishment cannot be foreseen and usually have a time limit for instantiation. The final reason is the diversity of topologies in the Internet. The algorithms presented in this paper address all of these issues except the proposed admission control.

Table 1 compares the characteristics of the algorithms proposed in this paper with those of the existing algorithms summarized in this section. The columns of the table list some of the characteristics that should be considered by an ideal mapping algorithm, while the rows represent the characteristics of the algorithms presented in the literature.

Table 1 shows that the number of router processing cores and the bandwidth of the links is being considered by the most of the algorithms. However, our work is unique since it considers: sets of images with different sizes, the time required to instantiate virtual routers, the locations of the repository in which images are stored and the available memory of the physical routers. Restrictions on the usage of physical routers by virtual routers (locality restriction) is rarely accounted for, although it is quite important. Other characteristics such as link delay and the time threshold for instantiations of virtual networks are neglected by all previous papers. Therefore, our algorithms significantly improve the state of the art for mapping virtual networks onto substrate networks, since they provide a more realistic assessment of operational networks.

Our work does not impose any alignment constraints between virtual topologies and physical topologies. It is possible that the topology of physical routers and links allocated to a virtual network will be the same of that of the requested virtual network, but this happens only if the topology is the one which minimizes the bandwidth allocated. Moreover, such an alignment is not necessary to guarantee the QoS requirements of the application,

Table 1 Comparison of the algorithms

Reference	Number of processing cores	Bandwidth	Locality restrictions	Images for the virtual routers
[16]	no	yes	no	**no**
[5]	yes	yes	no	**no**
[4]	yes	yes	yes	**no**
[6]	yes	yes	no	**no**
[7]	no	yes	no	**no**
[15]	yes	yes	no	**no**
Our proposal	yes	yes	yes	**yes**
Reference	**Link delay**	**Available memory / size of images**	**Locality of repository of images**	**Instantiation time**
[16]	no	**no**	**no**	**no**
[5]	no	**no**	**no**	**no**
[4]	yes	**no**	**no**	**no**
[6]	no	**no**	**no**	**no**
[7]	no	**no**	**no**	**no**
[15]	no	**no**	**no**	**no**
Our proposal	yes	**yes**	**yes**	**yes**

which are indeed assured by the constraints of the mapping problem.

4 Proposed algorithms

The algorithms in this paper model requests dynamically arriving for virtual network establishment on network substrates. Each request specifies the topology of the virtual network, the resources demanded by the virtual network elements, and the QoS requirements, which include a time limit to instantiate it.

The proposed algorithms are based on 0-1 ILP formulations. One algorithm, called the Optimal algorithm, uses the exact solution of the formulations to define the mappings. The other algorithms, called approximated algorithms, employ relaxation techniques to reduce the time needed to find a solution for the formulations. Before presenting the algorithms, we will present the ILP formulations. This formulation differs from that in our previous work [9] since a two step approach has been introduced which reduce memory demands.

The following notation is used for the formulations of the problem:

- $N \subset \mathbb{Z}$ is the set of physical routers;
- $F \subset \mathbb{Z}$ is the set of physical links, with the physical link (n_1, n_2) connecting two physical routers n_1 and $n_2 \in N$;
- $M \subset \mathbb{Z}$ is the set of virtual routers;
- $V \subset \mathbb{Z}$ is the set of virtual links with the virtual link (m_1, m_2) connecting two virtual routers m_1 and $m_2 \in M$;
- $I \subset \mathbb{Z}$ is the set of images stored in the repository. Each image corresponds to a file with an operating system and a specific set of software ready to be instantiated in a physical router;
- $A \subset \mathbb{N}$ is the set of the number of available cores in the physical routers; $A(n)$, $n \in N$, gives the number of cores of router n;
- $P \subset \mathbb{N}$ is the set of the number of cores requested by the virtual routers; $P(m)$, $m \in M$, gives the number of cores required by the virtual router m to be instantiated;
- $C \subset \mathbb{R}$ is the set of values of the available bandwidth in the physical links; $C(f)$, $f \in F$, gives the available bandwidth in the link f;
- $Q \subset \mathbb{R}$ is the set of bandwidth values requested by the virtual links; $Q(v)$, $v \in V$, gives the bandwidth required by the virtual link v;
- $D \subset \mathbb{R}$ is the set of values of delays in the physical links; $D(f)$, $f \in F$, gives the delay in link f;
- $K \subset \mathbb{R}$ is the set of values of maximum delay allowed on a virtual link; $K(v)$, $v \in V$, represents the maximum delay allowed on the virtual link v;
- $L_{n,m} \in \{0, 1\}$ are the binary values that establish restrictions on locations. If the virtual router m can be mapped onto the physical router n, the value of the variable is 1. Otherwise, it is 0. This variable is useful for imposing policy restrictions related to the geographical location of routers.
- $R_{n,i} \in \{0, 1\}$ are the binary values that provide details about the location where images are stored. If the image i is located in a repository with a direct link dedicated to the physical router n, the value of the variable is 1. Otherwise, it is 0;
- $E_{m,i} \in \{0, 1\}$ are the binary values related to software restrictions. If the image i contains all the software requirements required by the virtual router m (operating system, protocol stacks, kernel modules and others), the value of the variable is 1. Otherwise, it is 0;
- $B \subset \mathbb{R}$ is the set of values that represents the memory available in the physical routers; $B(n)$, $n \in N$, represents the memory available in the router n;
- $G \subset \mathbb{R}$ is the set of image sizes; $G(i)$, $i \in I$, represents the size of the image i;
- $S \in \mathbb{R}$ is the time limit for instantiation of the VN;
- $T_{n,i} \in \mathbb{R}$ represents the time the physical router n takes to boot the image i;

The substrate network is represented by a graph (N, F) in which the physical routers are modelled as the vertices of the graph and the physical links as the edges. Similarly, the virtual network is represented by the graph (M, V).

Requests must specify the maximum delay allowed in the virtual network links (D and K), since this information affects the performance of network applications. The specific image to each virtual router must be defined because various configurations can exist (I and $E_{m,i}$). The content of each repository must be known ($R_{n,i}$) because this affects the path chosen to transfer the images. The size of the images should be considered because the routers have limited storage capacity (B and G). Moreover, it is important to consider that clients can have specific policies that prevent the utilization of certain physical routers ($L_{n,m}$). Furthermore, the maximum time acceptable for the instantiation of the VN must be considered (S, D, K and $T_{n,i}$). To our knowledge, parameters related to transfer and the instantiation of software images (I, $E_{m,i}$, $R_{n,i}$, B, G and $T_{n,i}$) have never been taken into consideration in previous mapping algorithms proposed in the literature.

The solution to the problem is given by the binary variables:

- $X_{n,m,i}$: if the virtual router m is mapped onto the physical router n using the image i then this value is 1; otherwise, its value is 0;

- $Y_{n,u,w}$: if the physical path used by the virtual link w includes the physical link (n,u) , this value is 1; otherwise, it is 0;
- $Z_{n,u,m}$: if the physical link (n,u) is used to transfer the image requested by the virtual router m, this value is 1; otherwise, it is 0.

4.1 ILP formulations

All the algorithms proposed in this paper are based on two ILP formulations that must be sequentially executed. The first (ILP-Mapping) searches for the solution of the problem of mapping routers and links of VNs onto routers and links of the substrate. The second (ILP-Image) searches for routes in the substrate for transferring images from the repositories to the nodes in the substrate which will host the virtual nodes. The employment of two ILPs reduces the time needed to find solutions when compared to the execution time needed for our previous formulation, which try to find routes and allocate physical routers and links in a single ILP [9]. The reduction in execution time is mainly due to the reduction of the search space.

The ILP-Mapping algorithm is formulated as follows:

Minimize $\sum_{n\in N}\sum_{u\in N}\sum_{w\in V} Y_{n,u,w} \times Q(w)$ subject to the following 11 constraints:

$$\sum_{n\in N}\sum_{i\in I} X_{n,m,i} = 1 \quad (C1)$$
$$\forall m \in M$$

$$\sum_{m\in M}\sum_{i\in I} X_{n,m,i} \leq 1 \quad (C2)$$
$$\forall n \in N$$

$$\sum_{m\in M}\sum_{i\in I} P(m) \times X_{n,m,i} \leq A(n) \quad (C3)$$
$$\forall n \in N$$

$$X_{n,m,i} = 0 \quad (C4)$$
$$\forall n \in N, \forall m \in M, \forall i \in I | L_{n,m} = 0 \text{ or} E_{m,i} = 0$$

$$\sum_{w\in V} Y_{n,u,w} \times Q(w) \leq C(w') \quad (C5)$$
$$\forall w' = (n,u) \in F$$

$$\sum_{n\in N}\sum_{u\in N} Y_{n,u,w} \times D(n,u) \leq K(w) \quad (C6)$$
$$\forall w \in V, (n,u) \in F$$

$$\sum_{m\in M}\sum_{i\in I} X_{n,m,i} \times G(i) \leq B(n) \quad (C7)$$
$$\forall n \in N$$

$$Y_{n,u,w} = 0 \quad (C8)$$
$$\forall n,u \in N, \forall w \in V | (n,u) \notin F$$

$$\sum_{u\in N} Y_{n,u,w} - \sum_{u\in N} Y_{u,n,w} = \quad (C9)$$
$$\sum_{i\in I} X_{n,m,i} - \sum_{i\in I} X_{n,a,i}$$
$$\forall w = (m,a) \in V, \forall n \in N$$

$$X_{n,m,i} \in \{0,1\} \quad (C10)$$
$$\forall n \in N, \forall m \in M, \forall i \in I$$

$$Y_{n,u,w} \in \{0,1\} \quad (C11)$$
$$\forall n,u \in N, \forall w \in V$$

The objective function of the ILP-Mapping algorithm minimizes the bandwidth allocated to requests for the establishment of a virtual network. By doing so, the formulation maximizes the bandwidth available for future requests.

The constraint ($C1$) establishes that each virtual router is allocated to a single physical router and that a single image is used to instantiate it. Constraint ($C2$) limits the number of virtual routers that can be allocated on a physical router per request, with only a single virtual router can be allocated to a given physical router per request. The constraint ($C9$) ensures that the set of physical links on which a virtual link is mapped constitutes a valid path. This constraint compares the in-degree and the out-degree of each physical router n. The constraints ($C3$) and ($C7$) express the limitations of the physical routers related to the number of cores and the amount of memory, respectively.

The constraint ($C4$) guarantees that the virtual routers will be instantiated using images that satisfy all software requirements as well as any geographic location defined by the client requesting the VN.

The constraints ($C5$) and ($C6$) express the limitations of the physical links. The constraint ($C6$) establishes that the total delay in the physical path allocated to a virtual link does not exceed the delay threshold requested for that virtual link. Constraint ($C8$) guarantees that only existing physical links can be used in the mapping of virtual links.

Constraints ($C10$) and ($C11$) define the domains of the variables as {0,1}, i.e., the variables are binary. If the value of these variables is 1, a router (or link) is allocated to a virtual router (or link). Otherwise, it is zero.

After the solution of the ILP-Mapping is found, the values of $X_{n,m,i}$ are used as input for the second formulation, entitled the ILP-Image formulation.

The ILP-Image is formulated as follows:

Minimize $\sum_{m\in M}\sum_{n\in N}\sum_{u\in N|(n,u)\in F} Z_{n,u,m} \times D(n,u) + \frac{Z_{n,u,m}\times G(i|X_{v,m,i}=1)}{C(n,u)}$ subject to the following 3 constraints:

$$\sum_{m\in M} Z_{n,u,m} = 0 \qquad (C12)$$
$$\forall n,u \in N|(u,u) \notin F$$

$$\sum_{v\in N} Z_{u,v,m} - \sum_{v\in N} Z_{v,u,m} = \qquad (C13)$$
$$X_{n,m,i} \times R_{u,i} - X_{n,m,i} \times (1 - \lceil \frac{|u-n|}{\alpha} \rceil)$$
$$\forall m \in M, \forall i \in I, \forall n,u \in N, \alpha = |N|$$

$$Z_{n,u,m} \in \{0,1\} \qquad (C14)$$
$$\forall n,u \in N, \forall m \in M$$

The objective function of the ILP-Image minimizes the time required to instantiate a VN. The time needed to instantiate each virtual router is the sum of the times required to transfer the image and to boot the operating system of the image. We assume here that two or more images can be transferred simultaneously on the same physical link.

Constraint ($C12$) guarantees that (u,v) will be used in the mapping only if it is a physical link in the substrate. The constraint ($C13$) establishes that the set of physical links allocated for the transfer of an image consists of a valid path in the substrate network. Constraint ($C14$) defines the domain of the variables.

The following subsections present the proposed algorithms. Subsection "Optimal algorithm" presents the algorithm that execute the implementation of the ILPs exactly as shown in this subsection. This algorithm is called the Optimal algorithm. Subsection "Root approximative algorithm" presents the Root Approximative algorithm which limits the search for a solution at an earlier stage than does the Optimal algorithm. Subsection "Algorithms based on relaxed versions of ILPs" presents four approximative algorithms based on relaxation technique. Relaxed versions of ILPs tend to find solutions faster than the original formulation of the problem. The approximative algorithms are called the Random Approximative algorithm, the Deterministic Approximative algorithm, Iterative Random Approximative algorithm and Iterative Deterministic Approximative algorithm. They differ from the algorithm employed to round off the real variable values to binary ones.

4.2 Optimal algorithm

The Optimal algorithm implements the two ILP formulations exactly as shown in Subsection "ILP formulations". To find the solution to the problem, it uses the Branch and Cut technique [18] which builds a tree with the root corresponding to the solution of a relaxed formulation of the original ILP and each node to a solution of the relaxed ILP formulation.

The search for the solution starts at the root of the tree and as long as an integer variable is associated with a fractional value in relaxed version, new constraints (cuts) to the formulation are added reducing the search space (adjusted polyhedron). The addition of new constraints branches on a fractional variable creating two new nodes (sub-problems) in the search tree.

The Optimal algorithm traverses all the nodes of the search tree. It is possible either to establish deadlines for execution time of the traversal or to establish stopping criteria based on the position of the node in the tree.

In our formulation, the ILP-Mapping formulation traverses all the nodes of the tree and returns a solution that minimizes the allocated bandwidth. The ILP-Image formulation solution also traverses all nodes and minimizes the VN instantiation time. The Optimal algorithm for solving the ILP formulations is presented in Algorithm 1.

Algorithm 1. Optimal algorithm

Data: Substrate network γ with characteristics α, virtual network δ with characteristics β.
Result: Mapping of δ on γ and on the physical paths θ used to transfer the images.

1: Define γ, α, δ and β as input of the ILP-Mapping;
2: Traverse the entire search tree of the ILP-Mapping and obtain the values of $X_{n,m,i}$ and $Y_{n,u,w}$ variables related to the best solution found;
3: **if** ILP-Image does not find any solution **then**
4: Block the request;
5: **end if**
6: **else**
7: Define γ, α, δ, β and $X_{n,m,i}$ variables as input to the ILP-Image;
8: Traverse the entire search tree of the ILP-Mapping and obtain the values of $X_{n,m,i}$ and $Y_{n,u,w}$ variables related to the best solution found;
9: **if** ILP-Image does not find any solution **then**
10: Block the request;
11: **end if**
12: **else**
13: Return the mapping of δ on γ using the values of the variables $X_{n,m,i}$ and $Y_{n,u,w}$;
14: Return the paths θ using the values of the variables $Z_{n,u,m}$.
15: **end if**
16: **end if**

The Optimal algorithm solves ILP formulations (lines 2 and 8) by passing the characteristics of both the virtual network and the substrate network (lines 1 and 7), and returns the mapping of virtual routers and links (lines 13 and 14) onto the substrate network. If no feasible solution is found, the request for VN establishment is rejected (lines 3, 4, 9 and 10).

Table 2 Characteristics of the algorithms based on the use of relaxed versions of the ILPs

Algorithm	# of times ILP-Mapping is executed	# of times ILP-Image is executed	Definition of variables
RAA	2	1	Draw considering probabilities
DAA	2	1	Higher value
IRAA	$\|M\| + \|V\|$	$\|M\|$	Draw considering probabilities
IDAA	$\|M\| + \|V\|$	$\|M\|$	Higher value

In the remainder of this paper, the Optimal algorithm will be referred as Opt.

4.3 Root approximative algorithm

Preliminary experiments with the Opt algorithm showed that it takes too long to find solutions involving substrates with more than 100 routers. This motivated us to implement an approximative algorithm, called the Root Approximative algorithm. This algorithm stops the traversal at the root of the tree. By doing that, a reduction on execution time is expected in comparison with the time required by the Opt algorithm. Such a criterion was derived from the observation that several solutions for the optimal problem are obtained at the root of the tree.

The Root Approximative algorithm differs from the Opt algorithm in solution to lines 2 and 8, which are respectively replaced by:

- **Line 2**: Stop the search for solutions to the ILP-Mapping at the root of the search tree and obtain the values of variables $X_{n,m,i}$ and $Y_{n,u,w}$;
- **Line 8**: Stop the search for solutions to the ILP-Image at the root of the search tree and obtain the values of variables $Z_{n,u,m}$.

In the remainder of this paper, the Root Approximative algorithm will be referred as Root.

4.4 Algorithms based on relaxed versions of ILPs

In addition to the Root algorithm, four other approximative algorithms are proposed. These algorithms relax integer constraints of the two ILP formulations in an attempt to reduce execution time. This relaxation replaces the constraints ($C11$), ($C12$) and ($C14$), by the constraints ($C11'$), ($C12'$) and ($C14'$), given below:

$$X_{n,m,i} \in \mathbb{R}_{[0,1]} \quad \forall n \in N, \forall m \in M, \forall i \in I \qquad (C11')$$

$$Y_{n,u,w} \in \mathbb{R}_{[0,1]} \quad \forall n, u \in N, \forall w \in V \qquad (C12')$$

$$Z_{n,u,m} \in \mathbb{R}_{[0,1]} \quad \forall n, u \in N, \forall m \in M \qquad (C14')$$

These new constraints modify the domain of decision variables from $\{0, 1\}$ to $\mathbb{R}_{[0,1]}$, so after finding the solution to the relaxed version of the ILP, it is necessary to round off fractional values to binary ones. The four algorithms differ in relation to the method implemented for this rounding off.

Each of the four algorithms consists of three steps: node mapping, link mapping and definition of paths for transfer of the required images. For each step two procedures defines: how to round the real values off to binary ones and when such rounding off should take place. In node mapping, the first procedure defines which variable will be rounded off to 1 since only one $X_{n,m,i}$ can be set to 1 for each virtual node m. The second procedure determines whether or not all the other variables values should be rounded off to 1. There are two options for each of these

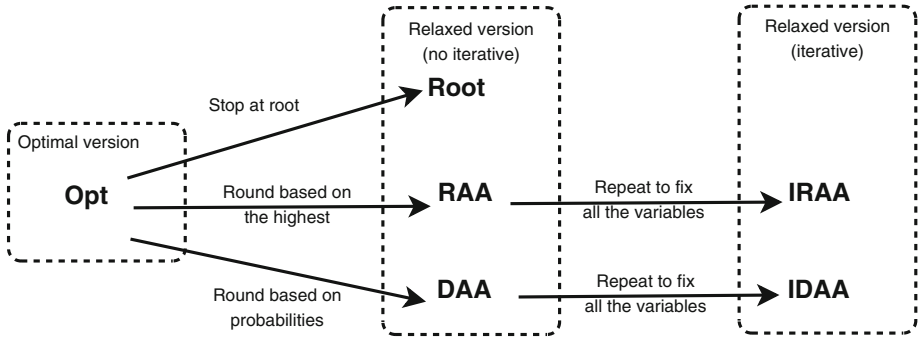

Figure 4 Summary of approximative algorithms.

Table 3 Types of virtual networks

Type	# of virtual routers	# of cores	Bandwidth (uniformly distributed)
1	5	2	100Mbps–200Mbps
2	8	3	200Mbps–300Mbps
3	10	6	300Mbps–400Mbps

two procedures with their combinations defining the four different algorithms proposed.

4.4.1 How to round off variables

The rounding off of real numbers can be either deterministic or random. In deterministic, the highest real value for a virtual node is rounded off to 1. In random algorithms, a random number is drawn and if this is lower than the value of the real variable, then the real value is rounded off to 1.

Such procedure is also employed for the Y and Z variables.

4.4.2 When to round off variables

After the execution of the relaxed ILP, another decision must be made. It is possible either to round all the X variables associated with all the virtual nodes at once or to round off only the X variables related to a specific virtual node, and then run the relaxed ILP as for each X variables. The same procedure applies to the Y and Z variables.

The option that round off all the variables at once implies two executions of the relaxed version of the ILP-Mapping. For the first, the value of the X variables are set and later used as input for setting the values of the Y variables. After the values of X and Y variables are set, the relaxed version of the ILP-Image is executed once to find the values of the Z variables. This is the procedure adopted by the non-iterative algorithms, i. e., the Deterministic Approximative Algorithm (DDA) and the Random Approximative Algorithm (RAA).

The other way is to set the value of a single variable after each execution of the relaxed ILP. In this case, the ILP-Mapping must be executed $|M|$ times to round off all X variables and another $|V|$ times to round off all the Y variables. The relaxed version of the ILP-Image must then be executed $|M|$ times to round all Z variables. This option is employed in the **Iterative** Deterministic Approximative Algorithm (IDAA) and in the **Iterative** Random Approximative Algorithm (IRAA). Table 2 summarizes the main characteristics of the four approximative algorithms proposed and Figure 4 illustrates the differences between all the algorithms presented in this section.

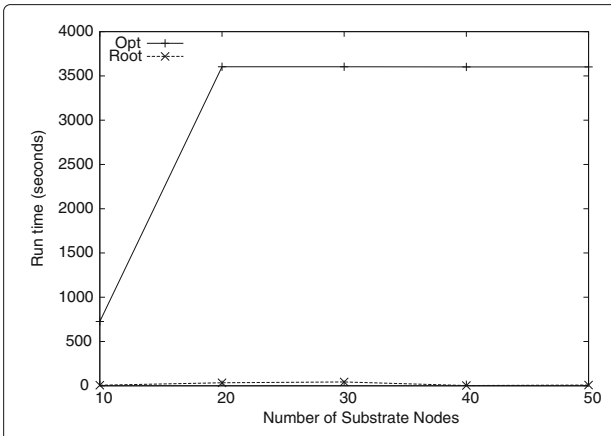

Figure 6 Run time for Type 2 requests in static scenarios.

5 Performance evaluation

This section assesses the efficiency of the proposed mapping algorithms. Numerical examples presented in this section compare the performance of the algorithms in both static and dynamic scenarios. The static scenario

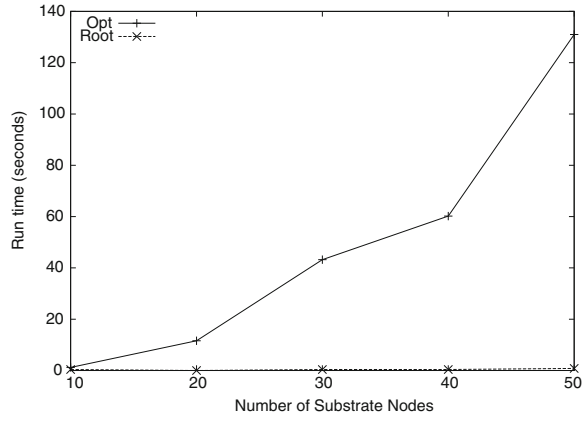

Figure 5 Run time for Type 1 requests in static scenarios.

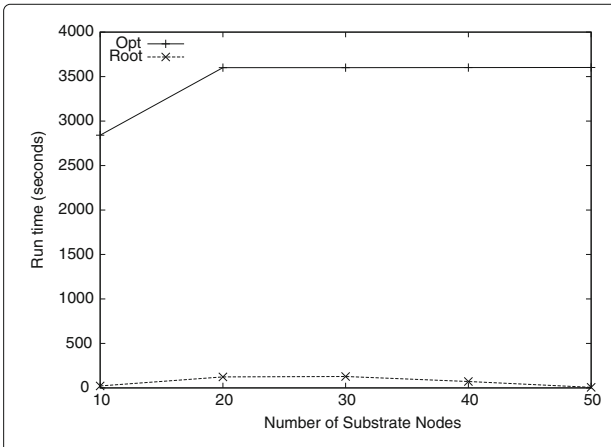

Figure 7 Run time for Type 3 requests in static scenarios.

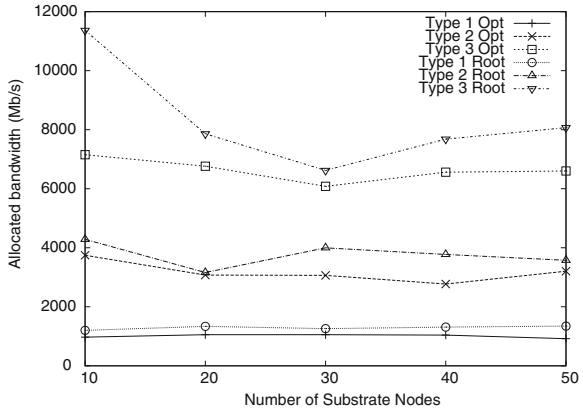

Figure 8 Bandwidth allocation for static scenarios.

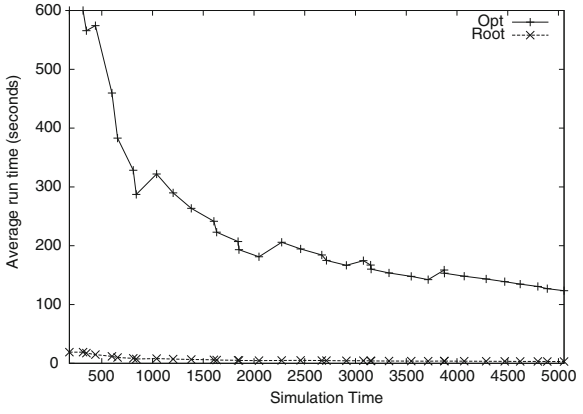

Figure 10 Run time for Type 2 requests in dynamic scenarios.

involves only the mapping of a single request. The dynamic scenarios involves requests arrive during a certain time interval, with the availability of resources in the substrate network varying over time. The algorithms were evaluated in terms of run time, the amount of bandwidth allocated to the virtual networks requests, and the blocking probability. A description of the experimental setup is followed by a comparison of the Opt and Root algorithms and another of the performance of the approximative algorithms. Comparisons with existing algorithms [4,19] were not performed, since these do not consider all of the parameters considered by the algorithms presented here.

5.1 Experimental setup

All the algorithms and the simulator were implemented in C++ with the linear program formulations implemented using the CPLEX optimization library version 12.0. All programs were executed on a computer running the operating system Debian GNU/Linux Squeeze. The computer was equipped with two Intel Xeon 2.27GHz processors each one with 6 cores capable of running 12 simultaneous threads and 40GB of RAM.

The configuration for the scenarios considered:

- Number of routers in the substrate network: 10 to 50. Using this variation, it is possible to evaluate the performance of the algorithms as a function of the number of physical routers. Moreover, for dynamic scenarios, the number of nodes in the substrate varied from 10 to 400 for the evaluation of the scalability of the approximative algorithms.
- Number of routers with attached image repositories set to 3. This value was experimentally found by the authors to avoid a large number of infeasible allocations;
- Number of cores available in the physical routers set to 6, which is the actual number found in real routers [20];
- Available bandwidth in real links determined by a uniform distribution between 1Gbps and 10Gbps,

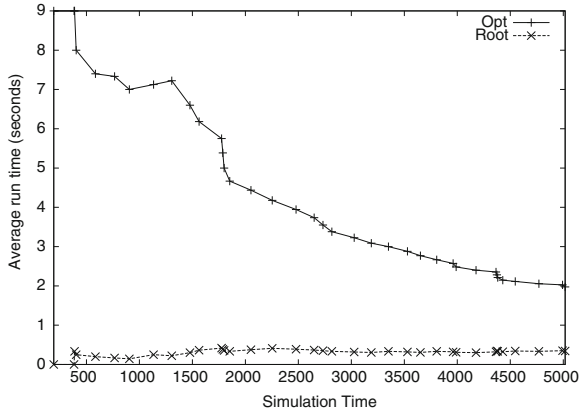

Figure 9 Run time for Type 1 requests in dynamic scenarios.

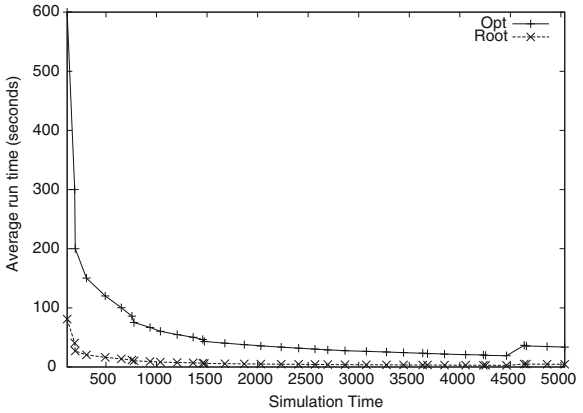

Figure 11 Run time for Type 3 requests in dynamic scenarios.

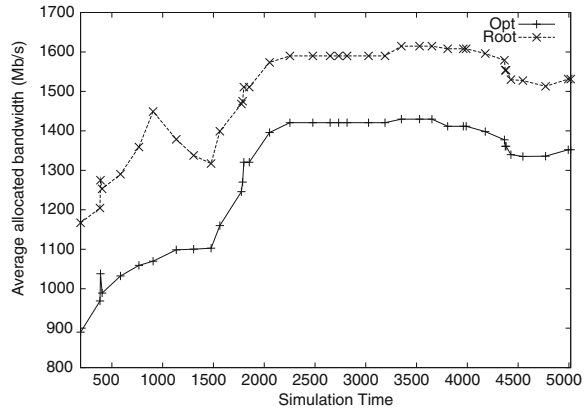

Figure 12 Allocated bandwidth for Type 1 requests in dynamic scenarios.

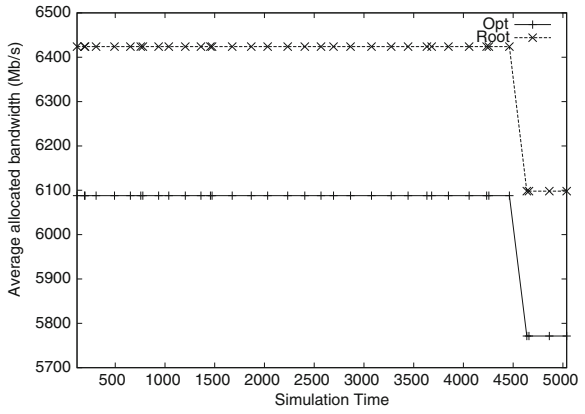

Figure 14 Allocated bandwidth for Type 3 requests in dynamic scenarios.

which is the interval common in substrate networks [21];

- Available memory in the physical routers set to 512MB; this number was based on the actual amount of flash memory in existing real routers [22];
- Size of images set to 128MB. This value was based on the amount of flash memory recommended for use of the software defined in [23], which is an operating system for routers;
- Time needed to boot an image in a physical router set to 10 seconds;
- Time threshold to instantiate each virtual network set to 100 seconds;
- Type of request: Type 1, Type 2 and Type 3. This depends on the number of resources required. Table 3 describes the requirements for each type of virtual network. They differ in terms of the number of requested virtual routers, the number of cores to instantiate each virtual router and in the guaranteed bandwidth per virtual link being requested. Requests are not known a priory; they are randomly generated in the ranges defined in the Table 3. The bandwidth demands for each request is defined in run time.

Both the topology of the substrate networks and that of the virtual networks were randomly generated by using the topology generator `BRITE` [24], with the BA-2 [25] algorithm, a method that generates network topologies similar to those found on the Internet. For the substrate network, the link delays were the values given by `BRITE`. Since the requested delay of the links of the virtual networks must be greater than those of the links of the substrate network, these were defined by multiplying the value returned by `BRITE` by a random number derived from a uniform distribution. For Type 1 virtual networks, the delay was calculated as the value given by `BRITE` multiplied by a number up to 15. For Type 2 virtual networks, the delay was calculated to be the value returned by

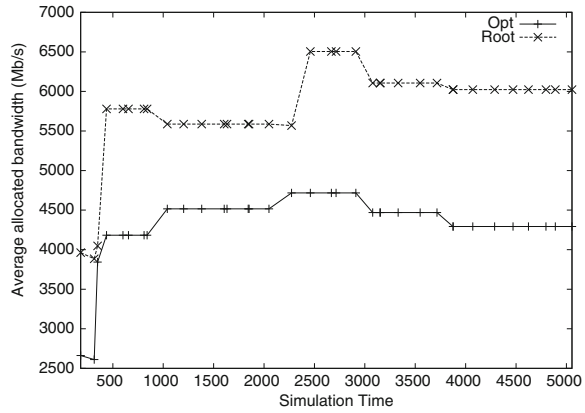

Figure 13 Allocated bandwidth for Type 2 requests in dynamic scenarios.

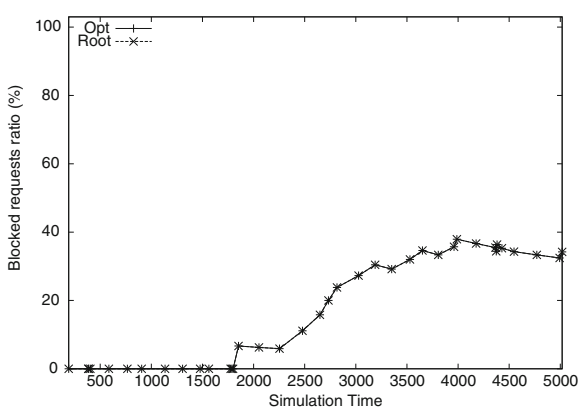

Figure 15 Blocked requests ratio for Type 1 requests in dynamic scenarios.

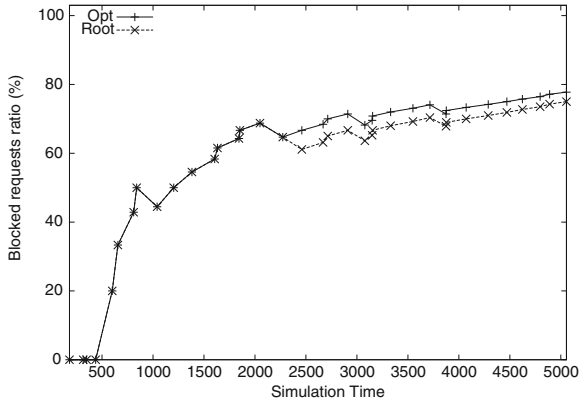

Figure 16 Blocked requests ratio for Type 2 requests in dynamic scenarios.

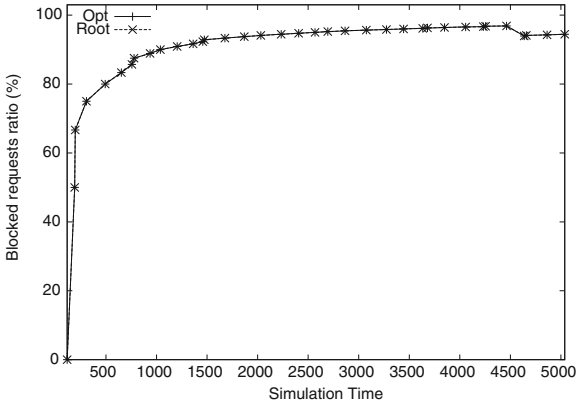

Figure 17 Blocked requests ratio for Type 3 requests in dynamic scenarios.

BRITE multiplied by a number up to 10. For Type 3 virtual networks, the delay was the value returned by BRITE multiplied by a number up to 5.

5.2 Optimal and root approximative algorithms

5.2.1 Static scenarios

The static scenarios involved only a single request, since the aim was to evaluate the differences between the proposed algorithms. The mapping of each request deals with an unallocated substrate. In this way, restrictions due to previous allocation have no impact on the difference of performance of the algorithms.

Results are reported as a function of the substrate size to evaluate the impact of it on the solution derived. The execution time of the Opt algorithm is limited to 3600 seconds. Each point in the graphs corresponds to the mean derived from five different requests.

Figures 5 and 6 plot the execution time of the algorithms as a function of the number of physical routers for requests of Type 1 and 2, respectively. For requests of type 1 (Figure 5) the execution time of the Root algorithm is less than that of the Opt algorithm with the execution time of the Opt algorithm increasing much faster than that of the Root algorithm as a function of the number of physical routers because of the increase in the search space. While the execution time of the Opt algorithm is 131 seconds for substrates with 50 nodes, the execution time of Root algorithm is less than 1 second. For requests of Type 2 which involve greater demands than do those of Type 1, while Root algorithm demands are still 6.4 seconds, the Opt demands 725.8 seconds for substrates with 10 nodes. Opt reaches the threshold for the execution time for substrates with only 20 nodes. For requests of Type 3 (Figure 7) the same trend is found although the execution time of the Opt algorithm was 2841 seconds for substrates with only 10 nodes.

Figure 8 plots the bandwidth allocated by the algorithms as a function of the number of physical routers. The Root algorithm always allocates more bandwidth than does Opt algorithm since only a limited number of solutions are evaluated. However, for requests of Type 1 the difference is quite small. For more demanding type of requests, these difference increases. For requests of Type 2, the maximum difference is 36.18%, while for Type 3 requests it is 58.91%.

These results show that the Root algorithm is more attractive than the Opt algorithm. The shorter run time of the Root algorithm and the similar bandwidth allocation justify the choice of the Root algorithm for the mapping on substrate with more than 30 nodes.

5.2.2 Dynamic scenarios

In the dynamic scenarios, several requests are included in each configuration of the network, so that the algorithms can be evaluated as the availability of the network changes as a function of time. The different sequence of resource allocations produced by different algorithms leads to different resource availability scenarios which implies different probabilities of success in the acceptance a request.

Simulation of each scenario took 5000 seconds. The arrival time and the duration of requests were defined randomly, on the basis of an exponential distribution with means of 100 and 2000 seconds, respectively.

Figures 9, 10 and 11 present the execution time of the algorithms for the three types of requests as a function of time. The execution time decreases along the simulation since resources are allocated and the search space shrink, as a consequence, the run time. Requests of Type 1 require low execution times since this type of requests can be easily accommodated. Although initially the difference is large, the execution time for the Opt algorithm is not

Table 4 Summary-dynamic scenarios

Opt			
Type	**Average run time (s)**	**Average allocated bandwidth (Mbps)**	**Average blocking rate**
1	4.48	1282.73	17.43%
2	245.80	4311.06	55.15%
3	71.77	6052.83	88.08%
Root			
Type	**Average run time (s)**	**Average allocated bandwidth (Mbps)**	**Average blocking rate**
1	0.30	1491.63	17.43%
2	6.46	5775.64	55.10%
3	9.84	6387.78	88.08%

that long. For requests of Type 2, differences in execution time are quite significant, being of the order of 600 seconds. For requests of Type 3 the differences are also large when the substrate is largely available, although the difference diminishes as the substrate becomes saturated. The average reductions in run time when using the Root algorithm were 99.93%, 99.97% and 99.86% for types 1, 2 and 3, respectively.

Figures 12, 13 and 14 show the bandwidth allocation per request. For requests of Type 1, the allocated bandwidth increases as the availability of resource decreases but reaches an almost constant value as the substrate occupancy tends to saturation. The greatest difference in bandwidth allocation was of 35.45%. The state of saturation is reached much faster as the demands of requests increases. For requests of Type 2 the maximum difference was in the order of 48.87%, while for Type 3, requests the bandwidth allocated per request by the Opt algorithm and Root algorithm was almost constant, equal to **6088** Mbps and **6424** Mbps, respectively.

Figures 15, 16 and 17 present results for the blocking ratio. As resources are allocated, their availability decreases leading to an increase in the probability of blocking. Requests of Type 2 and 3 saturate the substrate more quickly than do those of Type 1. The blocking rates for the two algorithms are very similar in despite of the difference of bandwidth allocated per request. This can be explained by the reduction of availability of physical routers leading to similar blocking ratio regardless of the savings in bandwidth.

Table 4 summarizes the results obtained in the dynamic scenario. Although the Root algorithm allocates in average 16.29%, 33.97% and 5.53% more bandwidth per request than does the Opt algorithm, these difference have no impact on the blocking ratio. In addition to yielding the same blocking ratio, the Root algorithm reduced the average run time in 99.93%, 99.97% and 99.86%, for types 1, 2 and 3, respectively. These results reinforce the advantage of the adoption of the Root algorithm given its reduced computational demand.

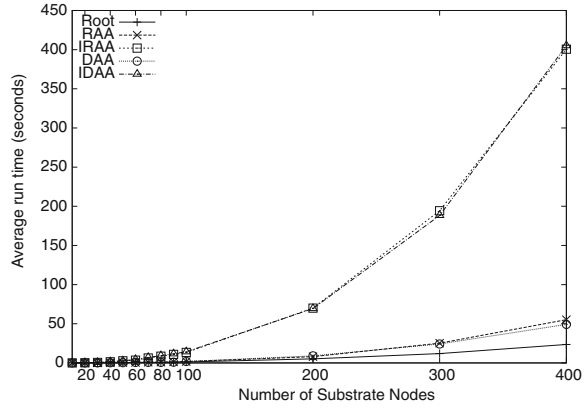

Figure 18 Run Time for dynamic scenarios.

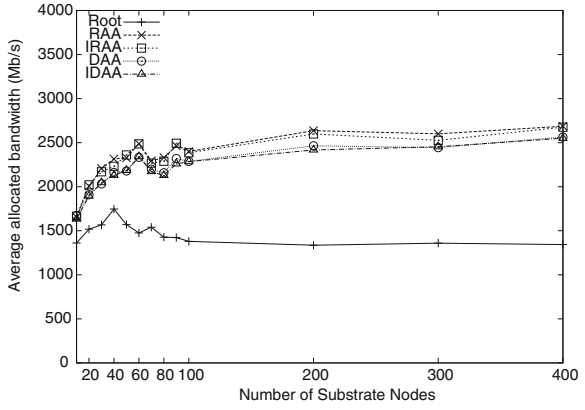

Figure 19 Allocated bandwidth for dynamic scenarios.

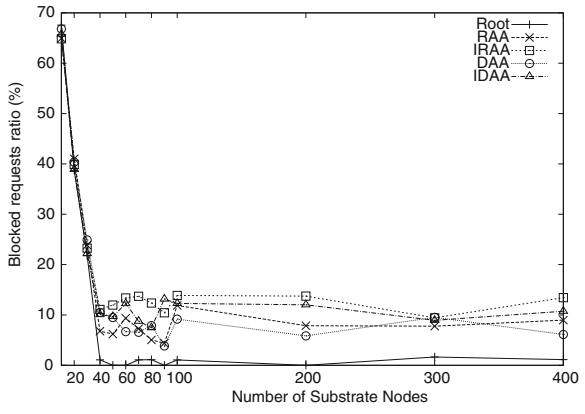

Figure 20 Blocked Requests Ratio for dynamic scenarios.

5.3 Approximative algorithms

The results produced by the approximative algorithms introduced in "Algorithms based on relaxed versions of ILPs" were compared to those yielded by the Root algorithm. In order to evaluate the growth in computational demands and the quality of the solution with an increase on the number of nodes in the substrate, the number of nodes in the substrate was up to 400 and the results are shown as a function of the number of nodes in the substrate.

Figure 18 shows that the average run time of the iterative algorithms (IRAA, IDAA) grows exponentially as a function of the number of nodes in the substrate and is ten times greater than that of the other approximative algorithms and twenty times greater than that of the Root algorithm for substrates with 400 nodes.

The iterative approximative algorithms allocate more bandwidth than do the other approximative algorithms and roughly 44.42% more than the Root algorithm (Figure 19). Moreover, the Root algorithm produces blocking ratio 8.93% lower than the other approximative algorithms as can be seen in Figure 20.

Table 5 summarizes the results found for the approximative algorithms. These results makes clear that the Root algorithm outperforms all other approximative algorithms. For instance, it requires 51.25 seconds less to run, on average, than does the IRAA and produces blocking ratio almost 8.93% lower.

Table 5 Numerical Comparisons (Average values)

	Type 1		
Algorithm	**Run time (s)**	**Bandwidth (Mbps)**	**Blocked requests**
Root	3.57	1314.57	10.40%
RAA	7.27	1816.10	15.79%
DAA	6.75	1706.33	15.97%
IRAA	54.82	1898.53	19.33%
IDAA	54.94	1827.07	17.86%

6 Conclusions and future work

Mapping virtual networks onto networks substrates is a crucial step for processing of VN services. therefore efficient mapping algorithms are of paramount for network virtualization.

This paper introduced six novel algorithms based on 0-1 ILP: one optimal and five approximative algorithms. These algorithms can be easily integrated to admission control mechanisms. They differ from previous proposals by the consideration of a large number of characteristics existing in real networks. It was shown via numerical examples that the Root algorithm demands considerably less computational time than the Opt algorithm and the iterative approximative algorithms. Such demand allows the adoption of Root algorithm for admission control in real time. It gives similar blocking ratio as does the Opt algorithm, and lower ratios than those of by the other approximative algorithms.

For future work, we intend to modify the formulation to consider the migration of virtual elements (routers and links), so that the algorithms potential migrations of VNs can be suggested. Formulations for the mapping problem considering path splitting are under development. We intent to verify results derived in a testbed for further validation.

Acknowledgements

This research was partially financed by Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), process 2010/03422-5.

Author details

[1]State University of Campinas, Campinas, Brazil. [2]University of São Paulo, São Paulo, Brazil.

Received: 4 December 2012 Accepted: 4 December 2012
Published: 30 January 2013

References

1. Zhu Y, Ammar M (2006) Algorithms for assigning substrate network resources to virtual network components. In: IEEE INFOCOM. INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, Barcelona, Spain, pp 1–12
2. Bless R, Hiibsch C, Mies S, Waldhorst O (2008) The Underlay Abstraction in the Spontaneous Virtual Networks (SpoVNet) Architecture. In: Next Generation Internet Networks (NGI 2008). Next Generation Internet Networks, 2008. NGI 2008, Krakow, Poland, pp 115–122
3. Feamster N, Gao L, Rexford J (2007) How to lease the internet in your spare time. SIGCOMM Comput Commun Rev 37(1): 61–64
4. Chowdhury N, Rahman M, Boutaba R (2009) Virtual Network Embedding with Coordinated Node and Link Mapping. In: IEEE INFOCOM. INFOCOM 2009. 28th IEEE International Conference on Computer Communications. Proceedings, Rio de Janeiro, Brazil, pp 783–791
5. Yu M, Yi Y, Rexford J, Chiang M (2008) Rethinking virtual network embedding: substrate support for path splitting and migration. SIGCOMM Comput Commun Rev 38(2): 17–29
6. Houidi I, Louati W, Zeghlache D (2008) A distributed and autonomic virtual network mapping framework. In: ICAS '08. Autonomic and Autonomous Systems, 2008. ICAS 2008, Gosier, Guadeloupe, pp 241–247

7. Lu J, Turner J (2006) Efficient mapping of virtual networks onto a shared substrate. Tech. Rep. W0UCSE-2006-35. Washington University , Washington, USA. http://www.arl.wustl.edu/jst/pubs/wucse2006-35.pdf. Accessed at 12/20/2010
8. Fan J, Ammar MH (2006) Dynamic topology configuration in service overlay networks: a study of reconfiguration policies. In: IEE INFOCOM. INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings, Barcelona, Spain, pp 1–12
9. Alkmim GP, Batista DM, Fonseca NLS (2011) Optimal mapping of virtual networks 2011. GLOBECOM '11. IEEE. In: Global Telecommunications Conference. Global Telecommunications Conference (GLOBECOM 2011), Houston, USA
10. Zhu Y, Zhang-Shen R, Rangarajan S, Rexford J (2008) Cabernet: Connectivity Architecture for Better Network Services. In: ACM CoNEXT '08. CoNEXT '08 Proceedings of the 2008 ACM CoNEXT, New York, USA, pp 64:1–64:6
11. Zhu Y, Bavier A, Feamster N, Rangarajan S, Rexford J (2008) UFO: a resilient layered routing architecture. SIGCOMM Comput Commun Rev 38(5): 59–62
12. Ricci R, Alfeld C, Lepreau J (2003) A solver for the network testbed mapping problem. SIGCOMM Comput Commun Rev 33(2): 65–81
13. Zhu X, Santos C, Beyer D, Ward J, Singhal S (2008) Automated application component placement in data centers using mathematical programming. Int J Netw Manag 18: 467–483. [http://dx.doi.org/10.1002/nem.707]
14. Padala P, Shin KG, Zhu X, Uysal M, Wang Z, Singhal S, Merchant A, Salem K (2007) Adaptive control of virtualized resources in utility computing environments. In: ACM EuroSys '07, pp 289–302
15. Botero J, Hesselbach X, Fischer A, de Meer H (2011) Optimal mapping of virtual networks with hidden hops. Telecommunication Systems 51: 1–10. doi:10.1007/s11235-011-9437-0
16. Szeto W, Iraqi Y, Boutaba R (2003) A multi-commodity flow based approach to virtual network resource allocation. In: Global Telecommunications Conference, 2003. GLOBECOM '03. vol 6. IEEE, San Francisco, USA, pp 3004–3008
17. Fletcher R, Leyffer S (1998) A mixed integer quadratic programming package . http://www.mcs.anl.gov/leyffer/solvers.html. Accessed at 02/22/2011
18. Gomory RE (1958) Outline of an algorithm for integer solutions to linear programs. Bull Am Soc 64: 275–278
19. Lischka J, Karl H (2009) A virtual network mapping algorithm based on subgraph isomorphism detection. In: ACM VISA '09. CM SIGCOMM 2009, Barcelona, Spain, pp 81–88
20. Cisco Systems (2010) Cisco Multiprocessor WAN Application Mode [Cisco Catalyst 6500 Series Switches]. http://www.cisco.com/en/US/prod/collateral/modules/ps5510/product_data_sheet0900aecd800f8965_ps708_Products_Data_Sheet.html. Accessed at 12/20/2010
21. RNP (2011) RNP Backbone map. http://www.rnp.br/en/backbone/index.php. Accessed at 09/19/2011
22. Cisco Systems (2011) Cisco 7200 Series Routers Overview [Cisco 7200 Series Routers]. http://www.cisco.com/en/US/prod/collateral/routers/ps341/product_data_sheet09186a008008872b.html. Accessed at 09/19/2011
23. Cisco Systems (2011) Download Software. http://www.cisco.com/cisco/software/release.html?mdfid=278807391&flowid=956&softwareid=280805680&release=12.4.2-XB11&rellifecycle=GD&relind=AVAILABLE&reltype=latest. Accessed at 09/19/2011
24. Medina A, Lakhina A, Matta I, Byers J (2011) Brite. http://www.cs.bu.edu/brite/. Accessed at 09/19/2011
25. Albert R, Barabási AL (2000) Topology of Evolving Networks: Local Events and Universality. Phys Rev Lett 85(24): 5234–5237

doi:10.1186/1869-0238-4-3
Cite this article as: Alkmim *et al.*: **Mapping virtual networks onto substrate networks.** *Journal of Internet Services and Applications* 2013 **4**:3.