Virtual Network Mapping – An Optimization Problem

Márcio Melo[1,2,⋆], Jorge Carapinha[1], Susana Sargento[2], Luis Torres[3], Phuong Nga Tran[3], Ulrich Killat[3], and Andreas Timm-Giel[3]

[1] Portugal Telecom Inovação, Aveiro, Portugal
{marcio-d-melo,jorgec}@ptinovacao.pt
[2] Instituto de Telecomunicações, University of Aveiro, Aveiro, Portugal
susana@ua.pt
[3] Institute of Communication Networks, Hamburg University of Technology, Hamburg, Germany
{luis.torres,phuong.tran,killat,timm-giel}@tuhh.de

Abstract. Network Virtualization is claimed to be a key component of the Future Internet by enabling the coexistence of heterogeneous (virtual) networks on the same physical infrastructure, providing the dynamic creation and support of different networks with different paradigms and mechanisms in the same physical network. A major challenge in the dynamic provision of virtual networks resides on the efficient embedding of virtual resources into physical ones. Since this problem is known to be $\mathcal{NP}$-hard, previous research focused on designing heuristic-based algorithms; most of them do not consider a simultaneous optimization of the node and the link mapping, leading to non-optimal solutions. This paper proposes an integer linear programming formulation to solve the virtual network embedding problem, as a simultaneous optimization of virtual nodes and links placement, providing the optimal boundary for each virtual network mapping. A $link - node$ formulation is used and the multi-commodity flow constrain is applied. In addition, a heuristic algorithm for virtual network embedding is also proposed and compared against the optimal formulation. The performance of the integer linear programming formulation and of the heuristic is evaluated by means of simulation. Simulation experiments show significant improvements of the virtual network acceptance ratio, in average additional 10% of the virtual network requests are accepted when using the integer linear programming formulation, which corresponds, in average, to more 7 virtual networks accommodated on the physical network.

Keywords: Embedding, ILP Model, Mapping, NP-hard, Optimization, Virtual Networks.

⋆ The author was supported by the Portuguese Foundation for Science and Technology (FCT) with a scholarship No. SFRH/BDE/33751/2009.

K. Pentikousis et al. (Eds.): MONAMI 2011, LNICST 97, pp. 187–200, 2012.

1 Introduction

Network Virtualization has gained an increasing prominence in networking and telecommunications fields in the last few years. Initially, the interest in network virtualization was mainly pushed by Future Internet research initiatives [1,2,3,4], mainly with the objective to find a platform on which novel Internet architectures could be experimented and evaluated without limitations or constraints, namely those associated with the traditional IP model. Later on, it became clear that virtualization could constitute a key component of next-generation Internet architecture itself [5], and not just as a mere platform for experimentation. Perhaps more importantly for network operators, it also became clear that network virtualization could provide a number of short/medium term business advantages, with potential reduction of costs and increase of revenues, as an interesting tool from an operational point of view [6,7]. Although there is a large interest on virtualized networks both from the research community and network operators, several challenges still prevent it from being deployed on real environments [8]. One of the major obstacles lies in the efficient embedding[1] of a Virtual Network (VN) onto a physical network. Since this process requires the simultaneous optimization of virtual nodes and links placement, it is complex in nature and requires large amounts of computing power. Some authors, such as [9,10,11,12,13,14,15], have already proposed solutions to this problem, mostly based on heuristic approaches, but have failed to provide the optimal solution for each VN mapping.

In this paper we propose a linear integer programming formulation to solve the VN assignment problem and to provide the optimal boundary for each VN embedding. The formulation supports heterogeneous virtual and substrate networks. In addition, we propose an heuristic algorithm based on [15] to solve the VN assignment problem. Simulation experiments show significant improvements of the VN acceptance ratio: in average more 10% of the VN requests are accepted when using the Integer Linear Programming (ILP) formulation, which corresponds to more 7 embedded VNs on the physical network. The paper starts with the discussion of the related work on existent mapping algorithms 2. Section 3 describes the network embedding problem, specifies the ILP model and shortly summarizes the enhancements proposed to a mapping heuristic based on [15]. Section 4 analyzes the performance of both the ILP optimization model and corresponding heuristic, and section 5 concludes the paper and describes the future work.

2 Related Work

This simultaneous node and link mapping optimization can be formulated as an un-splittable flow problem [9,16], known to be $\mathcal{NP}$*-hard*, and therefore, it is only tractable for a small amount of nodes and links. In order to solve this

[1] The terms embedding, mapping and assignment are used interchangeably in this paper.

problem, several approaches have been suggested, mostly considering the *off-line* version of the problem where the VN requests are fully known in advance.

In [10] a backtracking method based on sub-graph isomorphism was proposed; it considers the on-line version of the mapping problem, where the VN requests are not known in advance, and proposes a single stage approach where nodes and links are mapped simultaneously, taking constraints into consideration at each step of the mapping. When a bad mapping decision is detected, a back-track to the previous valid mapping decision is made, avoiding a costly re-map.

The work in [11] defines a set of premises about the virtual topology, i.e. the backbone nodes are star-connected and the access-nodes connect to a single backbone node. Based on these premises, an iterative algorithm is run, with different steps for core and access mapping. However, the algorithm can only work for specific topologies.

A distributed algorithm was studied in [17]. It considers that the virtual topologies can be decomposed in hub-and-spoke clusters and each cluster can be mapped independently, therefore reducing the complexity of the full VN mapping. This proposal has lower performance and scalability, when compared with centralized approaches.

Zhu et al. [9] proposed a heuristic, centralized, algorithm for dealing with VN embedding. The goal of the algorithm is to maintain a low and balanced stress of both nodes and links of the substrate network. However, the stress of nodes and links do not consider heterogeneity on their characteristics.

Yu et al. [12] propose an embedding algorithm which considers finite resources on the physical network, and enables path splitting (i.e. virtual link composed by different paths) and link migration (i.e. to change the underlying mapping) during the embedding process. However, this level of freedom can lead to a level of fragmentation that is infeasible to manage on large scale networks. In [13], it was taken a formal approach to solve the on-line VN embedding problem using a mixed integer programming formulation in a two-step approach. This approach, despite providing a better coordinated node and link mapping, it does not solve the VN assignment problem as an overall, and does not support heterogeneity of nodes.

Butt et al. [14] proposed a topology awareness heuristic for VN embedding and also suggest some algorithms to avoid bottlenecks on the physical infrastructure, where they consider virtual node reallocation and link reassigning for this purpose. Nogueira et al. [15] proposed a heuristic that takes into account the heterogeneity of the VNs and also of the physical infrastructure. The algorithm is evaluated by means of simulation and also on a small scale testbed, where it achieves mapping times of the order of tens of milliseconds.

Although all these algorithms provide a solution for the VN mapping problem, most of them fail to provide the optimal boundary for each VN mapping. Also, some of them fail to solve the assignment problem as a simultaneous optimization of the virtual node placement and of the virtual link placement, which lead to non-optimal solutions.

3 Problem Description and ILP Model Formulation

In this section, we start with the description of the VN assignment problem. The ILP model formulation is then presented, followed by a proposal of enhancement of a heuristic based on [15].

3.1 Virtual Network Assignment Problem Description

First, we start with the convention used for the index notation: i, j for nodes and links in the physical network, and m, n for nodes and links in the VN.

We consider that we have a physical network with a given number of nodes, N, and with a random topology, as depicted in figure 1. Each node is described by the number of Central Processing Unit (CPU), which correspond to letter C in the figure, the clock CPU frequency, F, and by the Random Access Memory (RAM) amount he possesses, M. With respect to the links, we consider the bandwidth capacity, B, and we assume that each link is an bi-directional link. Virtual networks are described the same way as physical networks, as shown in figure 2. We use the letter P when we want to refer to the physical resources, e.g. C^P, and the letter V is used for virtual resources, e.g. C^V.

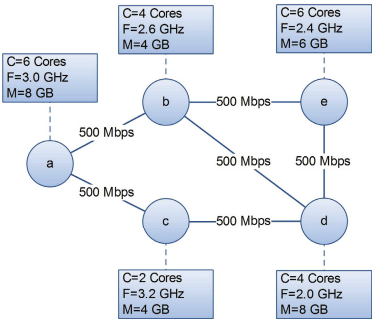

Fig. 1. Example of a substrate network topology description

The CPU capacity, the RAM size and the CPU frequency of the nodes is stored in a array with N^P entries, e.g. $C^P \rightarrow N^P \times 1$. We denote the total CPU capacity (the initial capacity) by $C^{P_{total}}$, the non-allocated capacity is denoted by $C^{P_{free}}$, and the used capacity is denoted $C^{P_{used}}$, where $C^{P_{total}} = C^{P_{used}} + C^{P_{free}}$. The same notation is used for the RAM. We use the adjacency matrix (1), $A^P \rightarrow N^P \times N^P$, to describe the connectivity of the physical network, and (2) to describe the connectivity of the VN, $A^V \rightarrow N^V \times N^V$.

$$A^P_{ij} = \begin{cases} 1, \text{ the physical node } i \text{ is neighbor of } j \\ 0, else \end{cases} \tag{1}$$

$$A^V_{mn} = \begin{cases} 1, \text{the virtual node } m \text{ is neighbor of } n \\ 0, else \end{cases} \tag{2}$$

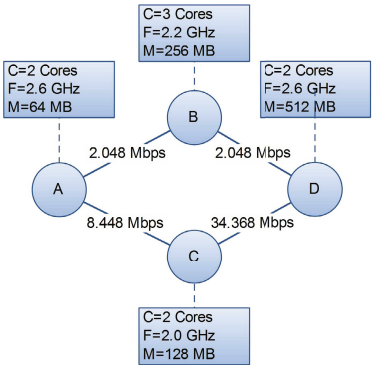

Fig. 2. Example of a virtual network topology description

3.2 ILP Problem Formulation

We use an ILP formulation [18] to solve the embedding problem of VNs . Here we only use two assignment variables for the VN mapping: for the virtual nodes, we use the binary variable x shown in equation (3), where $x_i^m \rightarrow N^V \times N^P$ matrix; for the virtual links, we use the binary variable y represented in equation (4), where $y_{ij}^{mn} \rightarrow (N^V)^2 \times (N^P)^2$ matrix (4-dimensional).

Our objective function is represented in equation (5) and is divided into two parts. Our primary goal is to minimize the maximum load per physical resource and, in the case of having different mapping solutions with the same maximum utilization, the second part of the objective function is activated which will opt for the solution that consumes the less physical links.

The maximum load at each different resource, i.e. memory RAM load (M_{load}), the CPU load (C_{load}), and the link load (B_{load}), is represented in equations (6),(7),(8), respectively. We multiply the CPU frequency by the CPU load in equation (6) and by the RAM load in equation (7), in order to firstly use physical nodes with lower frequency and to preserve the remaining for virtual nodes with higher frequency demands.

Equation (9) ensures that each virtual node is assigned and it is assigned to just one physical node, and equation (10) guarantees that each physical node can accommodate in maximum one virtual node per VN request, although each physical node can accommodate other virtual nodes from different VNs. We use equations (11) and (12) to make sure that we do not exceed the available capacity of all physical nodes, and we use equation (13) to guarantee that we do not violate that requirement on the CPU frequency.

In order to optimize the mapping of the virtual links and at the same time to cope with the optimization of the virtual nodes, we apply the multi-commodity flow constraint [19] with a $node - link$ formulation [20], and we also use the notion of direct flows on the virtual links, which are represented in equation (14). To ensure that we have enough bandwidth available at each physical link, we use equation (15).

Assignment Variables

$$x_i^m = \begin{cases} 1, \text{virtual node } m \text{ is allocated at physical node } i \\ 0, else \end{cases} \tag{3}$$

$$y_{ij}^{mn} = \begin{cases} 1, \text{virtual link } mn \text{ uses physical link } ij \\ 0, else \end{cases} \tag{4}$$

Optimization Function

$$minimize\ C_{load}^{max} + M_{load}^{max} + B_{load}^{max} + \epsilon \times \sum_{m,n \in N^V(m), n<m} y_{ij}^{mn} \times B_{mn}^V \tag{5}$$

Constraints

Derived from the Optimization Function

$$\forall i : F_i^P \times \frac{C_i^{P_{used}} + \sum_m x_i^m \times C_m^V}{C_i^{P_{total}}} \leq C_{load}^{max} \tag{6}$$

$$\forall i : F_i^P \times \frac{M^{P_{used}} + \sum_m x_i^m \times M_m^V}{M_i^{P_{total}}} \leq M_{load}^{max} \tag{7}$$

$$\forall i, j \in N^P(i), i < j : \frac{B_{ij}^{P_{used}} + \sum_{m,n \in N^V(m)} y_{ij}^{mn} \times B_{mn}^V}{B_{ij}^{P_{total}}} \leq B_{load}^{max} \tag{8}$$

Assignment of virtual nodes to physical nodes

$$\forall m : \sum_i x_i^m = 1 \tag{9}$$

One virtual node per physical node

$$\forall i : \sum_m x_i^m \leq 1 \tag{10}$$

CPU conservation

$$\forall i : \sum_m x_i^m \times C_m^V \leq C_i^{P_{free}} \tag{11}$$

Memory conservation

$$\forall i : \sum_m x_i^m \times M_m^V \leq M_i^{P_{free}} \tag{12}$$

Frequency limitation

$$\forall i : \sum_m x_i^m \times F_m^V \leq F_i^P \tag{13}$$

Multi-commodity flow conservation with $link - node$ formulation

$$\forall m, n \in N^V(m), m < n, \forall i : \sum_{j \in N^P(i)} (y_{ij}^{mn} - y_{ji}^{mn}) = x_i^m - x_i^n \tag{14}$$

Bandwidth conservation

$$\forall i, j \in N^P(i), i < j : \sum_{m,n \in N^V(m), m<n} B_{mn}^V \times (y_{ij}^{mn} + y_{ji}^{mn}) \leq B_{ij}^{P_{free}} \tag{15}$$

3.3 Mapping Heuristic Algorithm

In this section, we propose a heuristic algorithm for VN embedding, based on the one from [15]. A pseudo-code description of the mapping algorithm is shown in algorithm 1. With respect to the base algorithm, this one contains several changes.

First, we propose a different formula to determine the node stress, S_N, which is presented in equation (16). The formula used in the base algorithm to obtain the node stress, tends to balance the number of virtual nodes per physical nodes, in order to, favor nodes with lower CPU clock frequency, and to reduce the combination of consumed RAM and CPU. However, we could have physical nodes with different capacities and also virtual nodes with different requirements, which do not cope well with the objective of distributing the virtual nodes per physical nodes uniformly; moreover, physical nodes could be highly loaded at the CPU and mostly free at the RAM or the opposite, which, for the previously used formula was totally hidden, as long as, the combination of the two has the lower value. The node stress formula proposed, in equation (16) tends to balance the use of both RAM and CPU, at the same time, and also to favor nodes with higher clock CPU frequency.

Secondly, we added a tuning variable, β, which is used to tune the $link - path$ cost, $D(u, v)$, according to the neighborhood, reflected in lines 30 to 32. We have set the value of β to 0.01, which largely reduces the $link - path$ cost to virtual neighbors that have been already assigned.

As a third modification, we propose a new formula for calculating the node potential, i.e., π, shown in the lines 34 to 39. The formula proposed by Nogueira et al. [15] calculates the average to all possible destination nodes. Here, the node potential, is the average of the minimum $link - path$ cost to all the possible candidates to virtual neighbors, multiplied by the node stress, which corresponds to line 41.

The last improvement, is present in the lines 50 to 54 of the algorithm 1, where we dynamically update the link stress, S_{LS}, for physical links that have been already assigned to virtual links, during each VN mapping process.

$$S_{N_i} = \text{CPU\_Freq} \times [(\frac{M^{P_{used}}}{M^{P_{total}}})^2 + (\frac{C^{P_{used}}}{C^{P_{total}}})^2] \tag{16}$$

Algorithm 1. Mapping Algorithm Pseudo-Code

input : $Substrate$ (Substrate Network) , $V_{Request}$ (Requested VN)
output: V_{Map} (Mapped Virtual Network)

foreach *Link i in Substrate.Links* **do**
  foreach *VN j in Substrate.VNs* **do**
    foreach *Link k in j.Links* **do**
      if *Link* $k_j \supseteq$ *Link i* **then**
        $S_{LS}(i)$ += $S_{LV_j}(k_j)$;
      end
    end
  end
end
foreach *Link i in Substrate.Links* **do**
  $S_{LS}(i) = \sum_j^{N_V} \sum_k^{L_{V_j}} ((S_{LV_j}(k_j) | k_j \supseteq i))$;
end
foreach *Node i in Substrate.Nodes* **do**
  S_{N_i} = CPU_Freq $\times$ $[(\frac{M^{P}_{used}}{M^{P}_{total}})^2 + (\frac{C^{P}_{used}}{C^{P}_{total}})^2]$;
  $\pi(v) = 0$;
end
foreach *Node n in* $V_{Request}$*.Nodes* **do**
  foreach *Node i in Substrate.Nodes* **do**
    if *MeetsConstraints(n, i)* **then**
      n.Candidates.Add(i) ;
    end
  end
end
foreach *Node n in* $V_{Request}$*.Nodes* **do**
  foreach *Link k connected to n* **do**
    ConnectedVNode=GetLinkDestination(k) ;
    foreach *SourceCandidate v in n.Candidates* **do**
      foreach *DestCandidate u in ConnectedVNode.Candidates* **do**
        D(v,u)= Cost(CSFP_Dijkstra(v,u));
        if *u.Map* **then**
          D(v,u)=β $\times$ D(v,u);
        end
      end
      if $\pi(v)$ **then**
        $\pi(v) = mean[\pi(v), min(\forall u \in V_C : D(\text{v,u})]$;
      end
      else
        $\pi(v) = min[\forall u \in V_C : D(\text{v,u})]$;
      end
    end
    $\pi(v) = \pi(v) \times S_{N_v}$;
  end
  n.Map = $v : \pi(v) = min(\pi)$;
end
foreach *Node n in* $V_{Request}$*.Nodes* **do**
  $V_{Map}.Nodes \cup n$;
  foreach *Link k connected to n* **do**
    ConnVNode=GetLinkDestination(k) ;
    $V_{Map}.Links \cup$ CSFP_Dijkstra(n.Map,ConnVNode.Map) ;
    foreach *Link i in Substrate.Links* **do**
      if $V_{Map}.Links$ **then**
        $S_{LS}(i)+ = S_{LV_n}(k)$;
      end
    end
  end
end

4 Evaluation Results

In this section, we describe the simulation scenario and depict our major results. Our evaluation is primarily focused on the VN acceptance ratio according to different number of demands, i.e. average number of VN requests per time unit, and also on how many VNs can be accommodated on the physical network using the proposed model. We also compare the ILP model with the heuristic described in the previous section.

4.1 Simulation Parameters

In order to evaluate the ILP model according to different number of VN requests per time unit, we have implemented a discrete event simulator in Matlab®.

The physical network topology was created using the Waxman random topology generation method [21], and the number of physical nodes was set to 30. The recommended parameters for link probability, $\alpha = 0.4$ and $\beta = 0.1$, were used although some topologies did not have full connectivity, i.e. one physical node with no viable path to all the remaining nodes (e.g. a node with no links or non-connected clusters). In order to circumvent this, after generating the topology, additional links were added to the nodes with fewer interfaces, until total connectivity was reached. For each substrate node, a set of parameters was randomly attributed, from a pool of possible ones, using an uniform distribution, such as RAM amount, number of CPUs and CPU frequency. The physical link's bandwidth was set at a fixed bitrate. The set of parameters is presented on table 1.

The VNs were generated using the same topology generation model, although the number of virtual nodes was not fixed, but follows a uniform distribution, from 2 to 10 virtual nodes per VN topology. After generating the virtual topology, the same set of specifications was assigned, with a uniform distribution. The virtual nodes specification pool can be observed on table 2.

We assume that each VN request arrive according to a Poisson distribution and that each VN has an associated lifetime with an average of $\mu = 75$, following an exponential distribution. Regarding the average number of VN requests per time unit, we have started with 0.8 VN requests per time unit and we have increased by intervals of 0.2 until reaching 1.8. For each different demand, i.e. value of $1/\ \lambda$, 10 trials were performed. A new set of VN requests and a new physical network topology was generated for each trial and for each value of $1/\ \lambda$. All simulations were set to run until 1000 time units. A confidence interval of 95% is used for every result presented below.

We have used CPLEX®[22] version 11 to solve the linear programming problem, and a time limit of 600 seconds was defined for each VN mapping, although most VNs were embedded in hundred of milliseconds.

4.2 Simulation Results

We used several performance metrics to evaluate the optimal model and the heuristic algorithm. We measured the acceptance ratio and the number of

Table 1. Physical Nodes Pool Parameters

N. CPUs	{2; 4; 6}
CPU Frequency (GHz)	{2.0 to 3.2 in 0.2 steps }
RAM Memory (GB)	{2; 4; 6; 8}
Link Bandwidth (Mbps)	{500}

Table 2. Virtual Nodes Pool Parameters

N. CPUs	{1; 2; 3; 4 }
CPU Frequency (GHz)	{2.0 to 2.6 in 0.1 steps }
RAM Memory (MB)	{64; 128; 256; 512 }
Link Bandwidth (Mbps)	{2.048; 8.448; 34.368}

accommodated VNs as a function of the number of requests. We also measured the average memory RAM and CPU utilization on the nodes, and the average bandwidth utilization on the links. In all these cases, we plot the performance metrics as function of the number of VN requests per time unit.

Figure 3 presents the VN acceptance ratio of the proposed ILP model ('optimal') and of the enhanced heuristic ('heuristic') for different number of requests ('number of demands'). As can be observed, the acceptance ratio decays linearly with the number of requests for both mapping methods. This is expectable once we have more VNs to embed with the same amount of physical resources. The optimal method achieves a higher acceptance ratio, for instance, with a $1/\lambda = 0.8$, i.e. 0.8 VN request per time unit, nearly all (i.e. 95%) the VNs are accepted when using the ILP model, while with the heuristic only 85% of the requests are accepted.

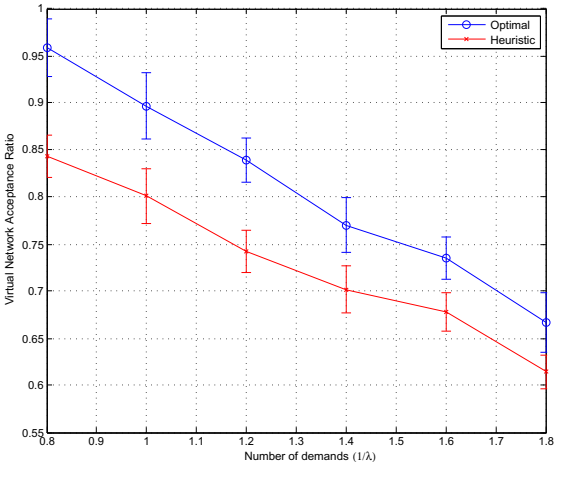

Fig. 3. Average acceptance ratio per demand

The average number of accommodated VNs per request is depicted in figure 4. For both embedding methods, there is a linear increase in VNs allocated in the substrate with the number of mapping requests. Note that the optimal model accommodates in average more VNs than the heuristic algorithm, approximately 7 VNs. We also observe that the number of VNs accommodated tend to a maximum value, which is expected since we have a finite amount of physical resources. For values of $1/\lambda \geqq 1.4$ we already realize this behavior, it is expected that the physical network would accommodate a maximum of 90 VNs using the optimal method and just over 80 when using the heuristic.

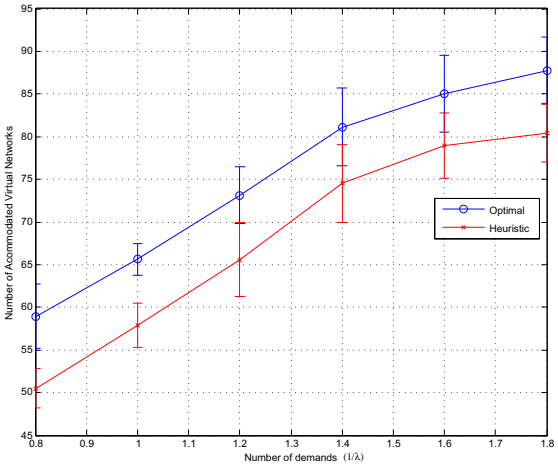

Fig. 4. Average number of accommodated virtual networks per demand

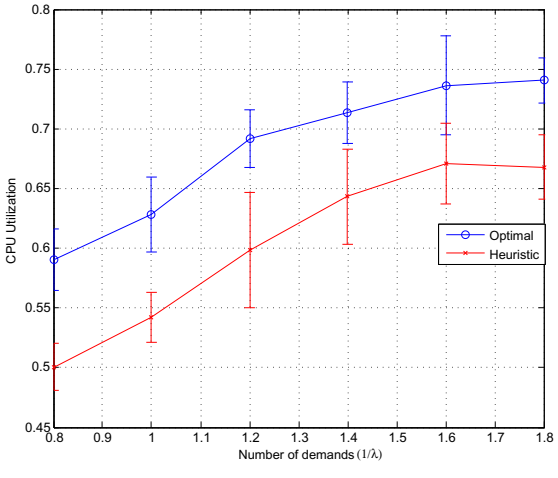

Fig. 5. Average CPU utilization per demand

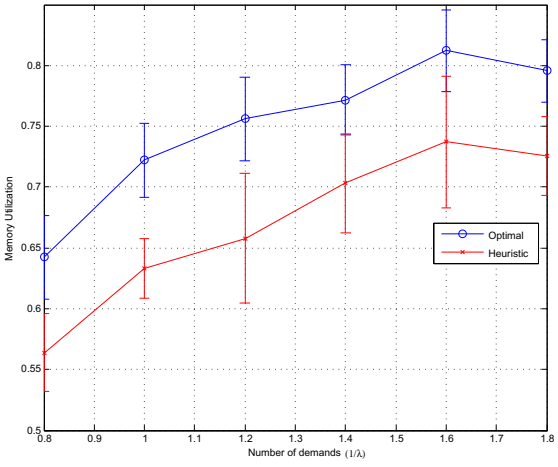

Fig. 6. Average memory utilization per demand

The remaining results concern the average utilization on the different types of resources: memory RAM and CPU on the physical nodes and bandwidth on the physical links, according to the VN requests. The resource utilization metric can be easily associated with the efficiency and is useful for two main reasons: (1) how much can the network operator load the physical resources; (2) what type of resources become scarce sooner. Figure 5 shows the average CPU utilization for different number of demands. Both embedding methods produce an increase of the CPU utilization with the number of requests, loading,

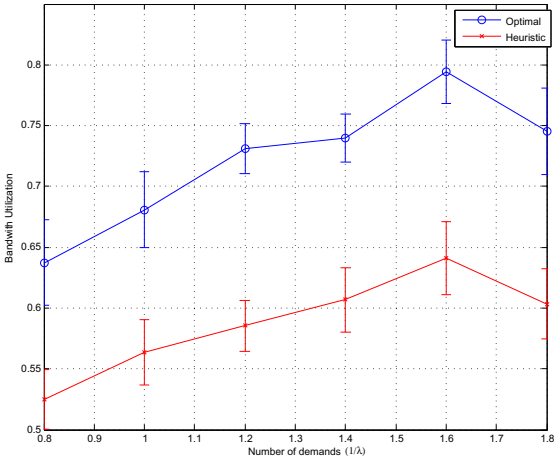

Fig. 7. Average bandwidth utilization per demand

in average, the CPU at a maximum of 74% and 62% for the ILP model and for the heuristic algorithm, respectively. The average memory RAM utilization according to different demands is depicted in figure 6. Both mapping methods produce an increase of memory utilization with the number of requests, reaching a stable value of 80% with the ILP model and 73% with the heuristic. The average bandwidth utilization is depicted in figure 7. Again, with both mapping methods, there is an increase of the resource utilization, with the optimal model reaching higher utilization values (i.e. 75%) and the heuristic method not going beyond 60%.

5 Conclusion

This paper proposed an ILP model to solve the VN embedding problem and to provide the optimal boundary for each VN mapping. The model applies optimization theory and simultaneously optimizes the virtual nodes and the virtual links assignment, supporting heterogeneous virtual and substrate networks. This paper also proposed an enhancement to a heuristic algorithm that is used as comparison with the ILP model.

The obtained results show significant improvements of the VN acceptance ratio, when we compare the ILP model with the heuristic. In average, the ILP model is able to map additional 10% VN requests. Translating this in the number of extra VNs accommodated on the physical network, in average, 7 more VNs are allocated in the substrate. The ILP model is able to load the physical resources to a maximum of 80% in average, with a high VNs demand, while the heuristic does not go beyond 74%.

Future work will endorse the global optimal solution (which may require reassignment of previously mapped virtual nodes or links), and the migration of virtual nodes and networks. Scalability issues of the proposed model will also be addressed.

References

1. Peterson, L., Anderson, T., Culler, D., Roscoe, T.: A blueprint for introducing disruptive technology into the internet. SIGCOMM Comput. Commun. Rev. 33, 59–64 (2003), http://doi.acm.org/10.1145/774763.774772, doi:http://doi.acm.org/10.1145/774763.774772
2. Anderson, T., Peterson, L., Shenker, S., Turner, J.: Overcoming the Internet Impasse through Virtualization. Computer 38, 34–41 (2005), http://portal.acm.org/citation.cfm?id=1058219.1058273, doi:10.1109/MC.2005.136
3. Feamster, N., Gao, L., Rexford, J.: How to lease the internet in your spare time. SIGCOMM Comput. Commun. Rev. 37(1), 61–64 (2007), http://doi.acm.org/10.1145/1198255.1198265
4. Zhu, Y., Zhang-Shen, R., Rangarajan, S., Rexford, J.: Cabernet: connectivity architecture for better network services. In: Proceedings of the 2008 ACM CoNEXT Conference, CoNEXT 2008, ACM ID: 1544076, pp. 64:1–64:6. ACM, New York (2008), doi:10.1145/1544012.1544076

5. Touch, J., Wang, Y.S., Eggert, L., Finn, G.: A virtual internet architecture. ISI Technical Report ISI-TR-2003-570 (2003)
6. Carapinha, J., Jimnez, J.: Network virtualization: a view from the bottom. In: Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures, pp. 73–80. ACM, Barcelona (2009), http://portal.acm.org/citation.cfm?id=1592648.1592660, doi:10.1145/1592648.1592660
7. Melo, M., Sargento, S., Carapinha, J.: Network Virtualisation from an Operator Perspective. In: Proc Conf. sobre Redes de Computadores - CRC (2009)
8. Chowdhury, N.M.K., Boutaba, R.: Network virtualization: State of the art and research challenges. IEEE Communications Magazine 47(7), 20–26 (2009), http://www.mosharaf.com/wp-content/uploads/nv-overview-commag09.pdf
9. Zhu, Y., Ammar, M.: Algorithms for assigning substrate network resources to virtual network components. In: Proceedings of 25th IEEE International Conference on Computer Communications, INFOCOM 2006, pp. 1–12 (2006)
10. Lischka, J., Karl, H.: A virtual network mapping algorithm based on subgraph isomorphism detection. In: VISA 2009: Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures, pp. 81–88. ACM, New York (2009), doi: http://doi.acm.org/10.1145/1592648.1592662
11. Lu, J., Turner, J.: Efficient mapping of virtual networks onto a shared substrate. Tech. rep., Washington University in St. Louis (2006)
12. Yu, M., Yi, Y., Rexford, J., Chiang, M.: Rethinking virtual network embedding: Substrate support for path splitting and migration. ACM SIGCOMM Computer Communication Review 38(2), 17–29 (2008)
13. Chowdhury, N., Rahman, M., Boutaba, R.: Virtual network embedding with coordinated node and link mapping. In: INFOCOM 2009, pp. 783–791. IEEE (2009), doi:10.1109/INFCOM.2009.5061987
14. Farooq Butt, N., Chowdhury, M., Boutaba, R.: Topology-Awareness and Reoptimization Mechanism for Virtual Network Embedding. In: Crovella, M., Feeney, L.M., Rubenstein, D., Raghavan, S.V. (eds.) NETWORKING 2010. LNCS, vol. 6091, pp. 27–39. Springer, Heidelberg (2010)
15. Nogueira, J., Melo, M., Carapinha, J., Sargento, S.: Virtual network mapping into heterogeneous substrate networks. In: IEEE Symposium on Computers and Communications, ISCC 2011 (2011), http://www.av.it.pt/mmelo/nogueira2011mapping.pdf
16. Andersen, D.G.: Theoretical approaches to node assignment (2002) (unpublished manuscript)
17. Houidi, I., Louati, W., Zeghlache, D.: A distributed virtual network mapping algorithm. In: IEEE International Conference on Communications, ICC 2008, pp. 5634–5640 (2008), doi:10.1109/ICC.2008.1056
18. Wolsey, L.: Integer programming. IIE Transactions 32, 273–285 (2000)
19. Even, S., Itai, A., Shamir, A.: On the complexity of time table and multicommodity flow problems. In: 16th Annual Symposium on Foundations of Computer Science, pp. 184–193 (1975)
20. Pióro, M., Medhi, D., Service), S.O.: Routing, flow, and capacity design in communication and computer networks. Citeseer (2004)
21. Waxman, B.: Routing of multipoint connections. IEEE Journal on Selected Areas in Communications 6(9), 1617–1622 (1988), doi:10.1109/49.12889
22. IBM ILOG Optimization Products, http://www-01.ibm.com/software/websphere/products/optimization