Multi-Domain Virtual Network Embedding with Coordinated Link Mapping

Shuopeng LI, Mohand Yazid SAIDI, and Ken CHEN

L2TI, Institut Galilée, Université Paris 13, Sorbonne Paris Cité, Villetaneuse, France

{li.shuopeng, saidi, ken.chen}@univ-paris13.fr

Abstract—**Network Virtualization, which allows the co-existence of various logical networks on shared physical infrastructure, has become popular in recent years. The optimal mapping of virtual resource to physical resource is a major issue in network virtualization. This problem, called virtual network embedding (VNE), has been well explored in the context of one physical domain, which is in practice operated by a single infrastructure provider (InP). However, the needs of virtual network (VN) is rapidly growing, and quite a number of VNs have to be established across multi-domain. For multi-domain VNE, infrastructure providers can no longer just solve their own single domain VNE problem, but have to cooperate to build the whole VN. Therefore, new challenge arises for the multi-domain VNE, compared to traditional single domain VNE. The existing investigations on this problem mainly focus on decomposing a VN to sub VN for each domain, but little attention has been paid to the joint relation between intra-domain and inter-domain (peering) links. In this paper, we propose a multi-domain link mapping method which combines the intra and peering link mapping so as to optimize the overall resource utilization. Our approach is easy to be deployed since it is based on current Internet architecture. Evaluation shows that our approach brings improvements related to existing methods.**

I. Introduction

Network virtualization [1] is regarded as a solution to overcome some weakness of traditional network architecture. It makes easy to support various separated logical networks running over shared physical network. In virtualized network architecture, the service provider (SP) creates and manages virtual networks (VN) for end users,while infrastructure provider (InP) deploys the substrate network (SN) equipment and offers the necessary physical resources.

An important step of network virtualization is to establish VNs above SNs. This is referred as *virtual network embedding* (VNE). The VNE problem aims to find a mapping from the VN to SN in a way that objectives (*e.g.* cost) are optimized and constraints (*e.g.* bandwidth) are satisfied.

Large networks in current Internet architecture are organized by autonomous system (AS). An AS is one or several physical networks controlled by a single authority. In this article, we use the vocabulary of "*domain*" to denote the whole substrate network under the control of a single InP.

As VNs are getting more and more deployed, VNs in multi-domain will be more and more considered by potential VN users. Establishment of multi-domain VN is more difficult than the one on single domain for at least two reasons:

- First, a single domain VNE problem is mainly solved by linear programming (LP). If we had a complete vision of all the domains, a multi-domain VNE could be considered as a single domain VNE with a very large domain, so computationally harder to solve.
- More importantly, for various reasons (technical, commercial, etc.), the acquisition of full information in multi-domain is costly and often not possible. Only limited information is exchanged between InPs via protocols like BGP, so single domain approach cannot be re-used.

To address these challenges, multi-domain VNE frameworks are developed. Usually, there is a VN decomposition step followed by local sub VN mapping in each InP. Some authors introduced a broker-like additional actor, termed *Virtual Network Provider* (VNP) [2], between SP and InPs. The role of this VNP consists in assembling multi-domain information, decomposing VN and achieving the multi-domain VNE.

Existing solutions mainly focused on the decomposition of a multi-domain VN to each domain. One of the shortcomings in these frameworks is the lack of efficient link mapping method especially for the peering links which interconnect two domains.

In this paper, we propose an efficient framework of link mapping in multi-domain virtual network embedding context, which jointly consider the mapping of intra and peering links. In our approach, the peering links are mapped simultaneously along with intra domain links. Our approach is based on information usually disseminated by classical routing protocol (like BGP). Our simulation results prove that this solution results in better utilization of substrate resources.

The rest of this paper is organized as follows. Section II provides an overview of the related work. Section III presents our network model. Section IV presents our multi-domain VNE solution. The evaluation results are shown in section V. Section VI concludes this paper.

II. Related Work

A. Single domain VNE

The problem of single domain VNE is NP-hard [3] [4]. A basic off-line approach is proposed in [5], which performs the embedding in 2 stages (node then link mapping). The Multi Commodity Flow (MCF) is introduced in [6] to embed the virtual links The method in [7] takes into account the virtual links in node mapping stage. It privileges such node mapping that reduce the length of substrate link path.

Since 2-stage VNE solutions are lack of cooperation, some solutions mapping nodes and links in the same stage have been proposed. An approach based on subgraph isomorphism detection is presented in [8]. An other model in [9] applies the Markov Random Walk to rank nodes and then embeds links and nodes by using back-tracking strategy based on breadth-first search. In order to meet the change of requirements over time, dynamic VNE is proposed in [10] .

B. *Multi-domain information disclosure*

Because of politic and efficiency reasons, InPs can't disclose their complete information to others, so it is critical to make clear the information disclosure policy.

The proxy VNP could get peering links location and resource information but intra-domain links cannot be assumed to be available to VNP [11].

In [12], three types of resource information in each domain are provided to VNP:

- Node: its location, available capacity and unit price.
- Peering link: its vertices, available capacity and unit price.
- Intra-domain link: a length-based price for connecting any two nodes in its domain.

Based on the information disclosure policy above, we will describe our network model in the next section.

C. *Multi-domain VNE*

Multi-domain VNE framework can be decomposed into three major components [13]:

(i) partitioning the VN request into each InP via multi-domain node mapping method,
(ii) establishing inter-domain connection (peering links) between InPs using inter-domain paths,
(iii) embedding each sub VN request in each InP using intra-domain algorithm.

Based on multi-domain information model introduced in previous section, some centralized multi-domain VNE solutions are proposed in [14][11][12][15].

Many of them mainly focus on the *first component*. In [14], the authors introduce the cost of mapping a virtual node to a domain and the cost of mapping a link between two substrate nodes. Their node mapping algorithm optimizes the total embedding cost. The approach in [12] adopts the node mapping method [7] on a full-mesh topology which complies with partial information disclosure policy.

The *second component* is not very well explored compared to the first component. Existing solutions use simple policies to establish peering links. In [14], each peering link is considered as a single VPN (Virtual Private Network) connection. In [12], the flow of peering links is unsplittable between two domains, while the intra-domain sub virtual links are splittable. The peering link path is determined by Dijkstra's algorithm on VNP layer. Since VNP layer topology is not modified over time because of cost efficiency, Dijkstra's algorithm based peering links have always the same path. This phenomenon will result in difficulty of later intra-domain mapping. In [11], a virtual node is first mapped to substrate peering node to determine the peering link and the InP it belongs to. This approach is suitable for traffic matrix based VN [16] but not topology based VN.

Since establishing peering links is a part of link mapping, the chosen peering nodes will probably influence the intra-domain paths. We believe that there exist some inter-dependencies between the 2nd and 3rd components. To this end, we propose a framework which maps peering links jointly with intra-domain links in each InP.

III. Network Model

We adopt the usual substrate and virtual network model [7]. In addition, we describe VNP layer information based on existing multi-domain information model.

A. *Substrate Network*

A domain InP_i is modelled as an undirected graph $G_i^S(N_i^S, L_i^S)$, where N_i^S is the set of substrate nodes in domain i, L_i^S is the set of internal substrate links. Each substrate node $n_i^s \in N_i^S$ is associated with a CPU capacity $cpu(n_i^s)$ and a geographic location $loc(n_i^s)$. Each substrate link $l_i^s \in L_i^S$ is associated with a bandwidth capacity $bw(l_i^s)$.

Assuming that the substrate network covers K domains, there are some peering nodes (border nodes) which have peering links with other domains. The peering nodes set is denoted by N_i^{SP} ($N_i^{SP} \subset N_i^S$). The peering links between InP_i (*i.e.* G_i^S) and InP_j (*i.e.* G_j^S) is denoted by L_{ij}^S. We denote by $P_i^S = \cup_{j=1}^K L_{ij}^S$ the set of all of the peering links of InP_i, and by P^S with $P^S = \bigcup_{i=1}^K P_i^S = \bigcup_{(i,j)\in(1\ldots K)^2} L_{ij}^S$ the set of all of the peering links. The complete substrate network $G^S(N^S, L^S)$ is thus obtained as follows: $N^S = \bigcup_{i=1}^K N_i^S$, $L^S = (\bigcup_{i=1}^K L_i^S) \bigcup P^S$.

B. *Virtual network*

The virtual network is also modelled as an undirected graph $G^V(N^V, L^V)$, where N^V is the set of virtual nodes and L^V is the set of virtual links. Each virtual node $n^v \in N^V$ is associated with a CPU capacity demand $cpu(n^v)$, a geographic location $loc(n^v)$ and a distance constraint $dis(n^v)$ specifying how far a virtual node n^v can be placed from its $loc(n^v)$. Each $l^v \in L^V$ is associated with a bandwidth demand $bw(l^v)$. In addition, each virtual network G^V has a lifetime $t(G^V)$.

C. *VNP layer model*

VNP collects information provided by InPs. We assume that InPs provide exact information about their nodes, as well as the peering links. On the contrary, there is no exact information about the internal organisation of a domain. Similar to the existing solution [12], we assume that this information is given by InP for each couple of <node, peering node>, as if there was a *pseudo* direct link between these two nodes. Denote the

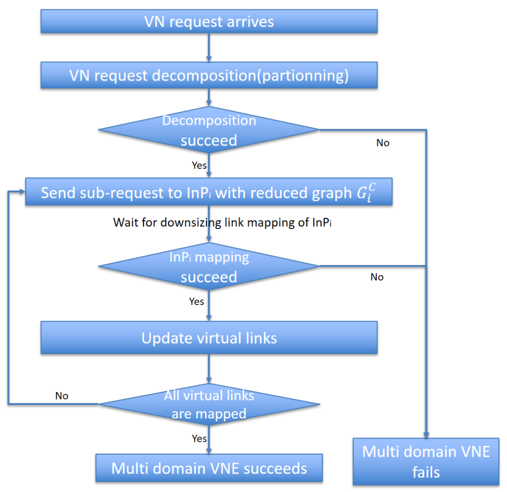
VN request arrives
VN request decomposition(partionning)
Decomposition succeed
No
Yes
Send sub-request to InPi with reduced graph G_i^C
Wait for downsizing link mapping of InPi
InPi mapping succeed
No
Yes
Update virtual links
No
All virtual links are mapped
Yes
Multi domain VNE succeeds
Multi domain VNE fails

Fig. 1: VNP workflow to embed a VN

set of these links by $L_i^P = \{l_{mn} \;/\; m \in N_i^S, n \in N_i^{SP}\}$, InP_i provides to VNP the set of linking cost C_i^P defined by

$$C_i^P = \{C(l_{mn}) \;/\; m \in N_i^S, n \in N_i^{SP}\}$$

where $C(l_{mn})$ represents a cost (distance, bandwidth, etc.) characterizing the link l_{mn}. This kind of information is actually what a routing protocol (BGP) reports to other AS.

Thus, the SN of an InP_i is *perceived* by VNP as a graph $G_i^P = (N_i^S, L_i^P)$. In this way, the whole substrate network that VNP perceives, referred as G^P, is defined as follows:

$$G^P = (\bigcup_i G_i^P) \bigcup P^S$$

i.e. the perceived vision for each domain and the exact vision of the inter-domain connections. With G^P, VNP can establish a kind of complete topology covering all the domains for achieving VN decomposition and link mapping.

IV. Our Proposition

To solve VNE in the context of multi-domain, we propose a novel algorithm that maps jointly intra and peering links.

We propose to handle each VN request with a 2-step process

- At the first step, VNP performs the node decomposition optimizing the node embedding.
- Subsequently, VNP performs a series of iterative downsizing VNE sub-solution, each of them optimizes both the intra and peering link mapping related to a domain.

The link mapping is determined, at each iteration, by the acting InP (called *mapper*). VNP is in charge of providing necessary information to the mapper. The generic work-flow of our algorithm is given by figure 1. The details are explained as below.

A. Decomposition

Firstly, VNP decomposes the VN request with objective of minimizing the node mapping cost. In this stage, VNP associates each virtual node with a candidate set of substrate nodes that meet its $loc(n^v)$. VNP is free to use any multi-domain VN partitioning method (*e.g.* [14][12]). At the end of this stage, virtual nodes are embedded to different domains.

An example of VN decomposition is shown in figure 2. Three InPs are shown with their substrate nodes from A to P. They are connected via 2 or 3 peering links. Intra substrate links are not drawn. We suppose that a VN $\{a, b, c, d\}$ arrives. The VN decomposition step tells us that a, b, c and d are mapped to substrate nodes A, K, N and J, respectively. $\{a-b, a-c, c-d\}$ are virtual links which interconnect two different domains, while $\{b-d\}$ locates in only one domain.

B. An iterative downsizing VNE approach

Here we give a detailed presentation of the kernel of our proposal, which is formally given in algorithm 1.

1) Rationale: After VN decompostition step, since there is no domain who knows the complete information of any other one, embedding the virtual links which interconnect two different domains becomes an issue.

We notice that, VNP can build, for each InP_i, a reduced vision (denoted by G_i^R) from G_i^P. This vision contains all the peering links/nodes, as well as the substrate nodes on which a virtual node is embedded. Formally, $G_i^R = (N_i^R, L_i^R)$ where

$$N_i^R = N_i^{SP} \bigcup \{n_i^S \in N_i^S \;/\; \exists n^v \in N^V, M(n^v) = n_i^S\}$$

i.e., N_i^R is the union of all the substrate nodes supporting virtual nodes on domain i and all of its peering nodes. In a similar way, we define L_i^R as follows;

$$L_i^R = \{l_{mn} \in L_i^P / n \in N_i^R, m \in N_i^{SP}\}$$

i.e., L_i^R is the subset of L_i^P between N_i^R and N_i^{SP} containing only the links interconnecting a peering node and a node supporting a virtual node.

In order to achieve an efficient and pragmatic operation mode, we prefer that VNP plays its role of coordinator: It is VNP who decides which of the InP should have the privilege to map its peering links with others. It is also VNP who provides to the chosen InP (that we refer as *mapper*) the topology of the rest of the network according to its perception. In other words, the chosen InP (the mapper) extends its view to the rest of the network, by using the vision provided by VNP, the only one who has a kind of comprehensive view on all domains. In this way, the mapper obtains an augmented graph on which it will perform link mapping, including both its intra and peering links.

This process continues, domain after domain, until all of the virtual links are set. The selection criterion is the link utilization, the InP has most stringent link utilisation will be the first to map its peering links. The reason lies in that high link utilization denotes more constraints in the choice of path.

2) Building of the augmented graph: Let InP_i be the chosen mapper. Formally speaking, the vision of the other domains provided by VNP is $G_i^C = \bigcup_{j \neq i} G_j^R$, i.e. the reduced perceived vision of all the other domains. We only need to consider the case where all the domains are adjacent to the

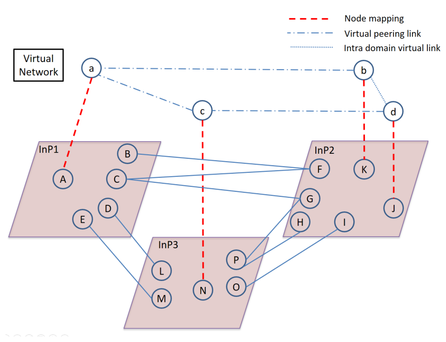

Fig. 2: VN decomposition

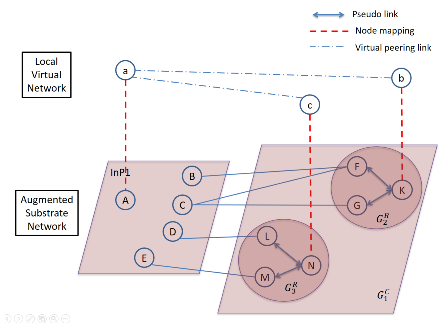

Fig. 3: downsizing link mapping by InP_1

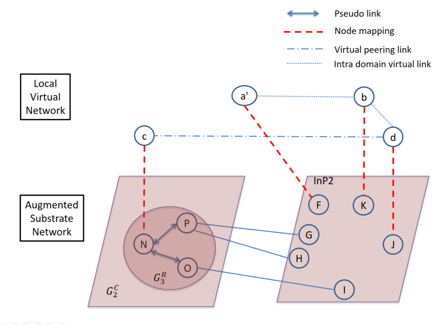

Fig. 4: downsizing link mapping by InP_2

mapper. The case of a domain not adjacent to the mapper but to which the mapper has virtual links can be reduced to the adjacent case.

VNP communicates G_i^C to the mapper (InP_i) so that the latter can creates an augmented graph G_i^A, defined as follows:

$$G_i^A = G_i^S \cup P_i^S \cup G_i^C$$

This topology covers all of the accessible domains and can be used as a substrate graph on which the mapper performs VNE.

3) VN sub-request: VNP asks the mapper to perform a partial VNE, which concerns only the virtual links related to the mapper. We refer this partial VNE as a *sub-request* (L_i^{subv}). It is obtained from the current VN request by reducing it to virtual links related to the mapper.

4) An MCF-based link mapping: At this stage, the mapper gets an augmented vision of the whole substrate network, and a VNE sub request (L_i^{subv}), both from VNP. We have thus a classical VNE problem that we solve with the multi commodity flow (MCF) based mapping algorithm (line 6 of algorithm 1).

At the end of this step, InP_i pre-allocates resources on the intra and peering links related to it and sends to VNP a virtual link update notification.

Let us illustrate it by our example. Assume in figure 2 that InP_1 is chosen as the 1st mapper. VNP builds the VNP-level graph vision $G_1^C = G_2^R \cup G_3^R$ (see figure 3) with G_2^R = ({*F*, *G*, *K*}, {*F-K*, *G-K*}) and G_3^R = ({*M*, *N*, *L*}, {*M-N*, *L-N*}). It builds also the sub-request L_1^{subv} = {*a-b*, *a-c*}, actually the virtual links *b-d* and *c-d* will be pruned since they haven't any extremity node supported by a substrate node in InP_1. VNP then sends G_1^C and L_1^{subv} to InP_1. The latter builds the augmented graph G_1^A which includes G_1^S (all the nodes and links in InP_1), the peering links (*B-F*, *C-F*, *C-G*, *E-M*), and G_C^1. InP_1 then applies the MCF-based algorithm to solve the embedding of L_1^{subv} on G_1^A.

C. Update and iteration

After each sub-request, the mapper (say InP_i) reports the results. In particular, it gives the results of the mapping of all of its inter-domain virtual links in the following manner.

Let $l^v(a,b)$ be the virtual link between a particular node $a \in InP_i$ and a particular node $b \in InP_j$, with $bw(l^v(a,b))$ as the required bandwidth. The MCF algorithm will map $l^v(a,b)$ into one or several paths. Denote by N^F the set of the peering nodes of InP_j through which a fraction of $l^v(a,b)$ is mapped. After the link mapping of InP_i, the set of virtual links $\{l^v(c,b)\}_{c \in N^F}$ is equivalent to the virtual link $l^v(a,b)$ with bandwidth demand:

$$\sum_{c \in N^F} bw(l^v(c,b)) = bw(l^v(a,b))$$

It is to be pointed out that these links are totally inside inP_j and they replace $l^v(a,b)$.

As the mapping of InP_i is achieved, it will no longer appear as domain in the subsequent problem which contains only the remaining domains. However, the achieved mapping concerns only the links related to InP_i (intra as well as peering), the part of inter-domain virtual links on the other domains still has to be mapped. Each of such inter-domain virtual link related to the mapper can be transformed into the above described equivalent set which will be added to each concerned domain. For the sake of reading simplicity, we prefer to give an informal description here, instead of a formal one, which would generate some more heavyly-indexed notations.

In this way, we obtain a new VNE problem with:

- at the SN level, the retreat of InP_i and all the peering links related to it;
- at the VN side, the retreat of all the virtual links internal to InP_i and the replacement of all the inter-domain virtual links related to InP_i by their equivalent set which are added to corresponding domain.

This allows us to execute iteratively the downsizing mapping described in § IV-B. VNP repeats the process till its convergence which is *certain*, since the subset is reduced by at least one domain (the mapper) at each iteration.

In the example of figure 2 and 3, assume that InP_1 has chosen link *F-K* to map virtual link *a-b*. After sending its results to VNP, this latter deduces and creates a new virtual link *a'-b* with node mapping *a'* equal to *F*. This virtual link *a'-b* replaces virtual link *a-b*.

Now, the new problem (figure 4) contains only InP_2 and InP_3. Assuming that the InP_2 is chosen as mapper, the same process continues and our problem is eventually reduced to a single domain which is the last step of our algorithm.

Algorithm 1: Link mapping of InP_i as mapper

```
   Input  : sub request virtual links $L_i^{subv}$
   Input  : reduced perceived graph $G_i^C$
   Output: virtual link update notification
1  begin
2      if $L_i^{subv} = NULL$ then
3          return
4      end
5      create augmented substrate network $G_i^A(N_i^A, L_i^A)$ ;
6      solve single domain VNE MCF problem;
7      foreach flow on substrate link $l_{mn}$ do
8          if $l_{mn} \in L_i^S \cup L_{ij}^S$ then  pre-allocate resource on
             link $l_{mn}$ ;
9      end
10     send virtual link update notification;
11 end
```

D. Reject of virtual request

The resources are definitively allocated only if all the computation on different domains succeed. A COMMIT message is then sent by VNP to InPs so as to validate the resource reservation. Should a mapper report a failure, a DEALLOC message would be sent by VNP, which stops the process (VNE failure) and allows each domain to deallocate pre-allocated resources.

V. Performance Evaluation

We implemented a discrete event simulator to evaluate the performances of our method. The optimization problem is solved by IBM CPLEX library. Since we are basically interested by the link mapping, all the evaluated methods work with the same node decomposition by using the greedy algorithm of [6].

A. Evaluation Environment

To simulate the multi-domain environment, we chose 4 real networks from SNDlib [17] interconnected via 44 peering links. The 4 domains consist respectively of 35 nodes and 80 links, 40 nodes and 89 links, 50 nodes and 88 links, and finally 54 and nodes 81 links. The CPU capacity of each node is chosen in [100,150]. The bandwidth capacity is selected in [100,150] for intra links and in [300,400] for peering links. The cost of the pseudo link between a border node and an intra node is chosen to be the inverse of the bandwidth capacity of the shortest path between these two nodes.

The virtual networks are generated by GT-ITM tool [18]. The virtual nodes of each VN follow a uniform distribution between 3 and 10. The virtual nodes are interconnected with probability 0.4. The CPU and bandwidth demands are uniformly chosen in [0,20]. The VN request arrival process is Poisson with arrival rate $\lambda \in (2 \ldots 6)$ requests per 100 time units. The life time of each VN follows an exponential distribution with an average of 1000 time units. Each simulation lasts for 30000 time units.

B. Compared methods

Our method, called $ciplm$ (Coordinated intra and peering link mapping), is compared against 3 different methods.

(i) $ideal$: The MCF link mapping is done with full information. This means that the multi-domain network is treated as a single domain. This is not feasible in practice.

(ii) $ciplm+$: This is a reinforcement of our method. It uses the real time attributes (residual bandwidth in particular) to compute the pseudo direct link cost. This method needs to update the residual bandwidth of each link in VNP every time a VN is accommodated. This method is not practical since it requires the exchange of a large amount of information and it claims a less strict information disclosure policy.

(iii) $shen$ [12]: this approach computes separately intra and peering links. The latter is determined according to Dijkstra's algorithm.

We used the following metrics for comparison:

- *VN request acceptance ratio*: the ratio of the accepted VN request over the total arrived VN requests;
- *Average link utilization*: average percentage of substrate link resource utilization over time;
- *Total revenue*: sum of VN demands (bandwidth, cpu).

C. Result analysis

The simulation results are shown in figure 5, 6 and 7. The VN request acceptance ratio is shown in figure 5. The link utilization and revenue are shown in figure 6 and figure 7, respectively. We got the following observations:

- $ideal$ is the best, followed by $ciplm+$, and our basic method $ciplm$ outperforms clearly $shen$ over all the three metrics.
- The difference between $ciplm$ and $shen$ is always significant.
- The difference between $ciplm+$ and $ciplm$ is not always significant, this is true in particular for the case of link utilisation (figure 6).

To summarize:

- Our approach is better than that of $shen$. Indeed, mapping jointly intra and peering links increases the efficiency. Our methods improves the performance in particular under heavy loads. In these cases, traffic is splitted and sent to less loaded links, achieving in this way a better utilisation of the overall residual bandwidth.
- The comparison between $ciplm$ and $ciplm+$ shows that the out-performance of the latter maybe be small. Considering the over-cost (in terms of information exchange and requirement on the information disclosure) of deploying $ciplm+$, we think that the $ciplm+$ does not offer a good trade-off between cost and performance.

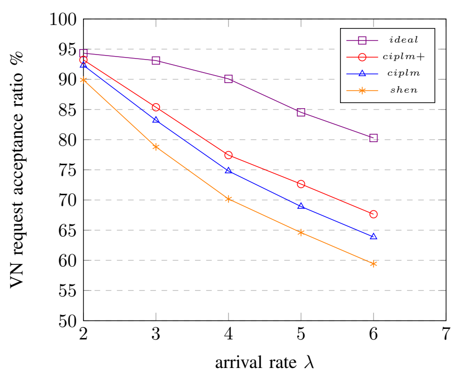

Fig. 5: Acceptance ratio

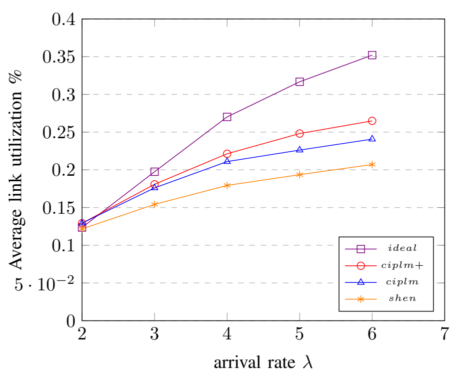

Fig. 6: Average link utilization

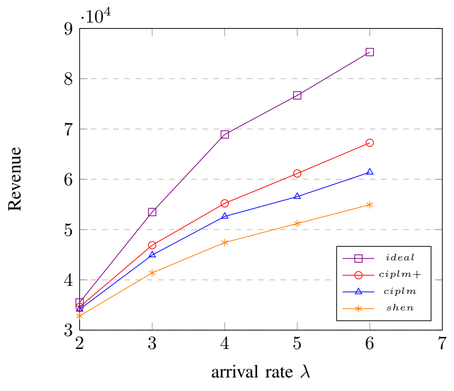

Fig. 7: Total revenue

VI. Conclusion

Network virtualization attracts more and more attention in future network architecture, since it allows the (dynamic) building of a network suited to end-users need, without modifying the underlay infrastructures. Part of them will be built over several infrastructures run by different operators.

The virtual network embedding, which aims at establishing the optimal virtual networks on substrate networks, is a key issue in network virtualization. The fact of partial information makes the multi-domain VNE quite different from the single-domain VNE and this problem remains a challenge. Some multi-domain VNE solutions have been proposed in literature. Most of them focus more on VN decomposition into sub VN requests for each domain, so that the single-domain VNE can be applied subsequently. Few attention has been paid on the mapping of peering (inter-domain) links.

In this paper, we propose a novel multi-domain VNE algorithm which aims to optimize the peering link mapping. For this, we introduce a coordinator (VNP, VN Provider). The latter has the privilege to get a comprehensive vision of all of the domains as well as the peering links. It performs VN decomposition, then coordinates the optimized mapping of both intra and peering links, domain after domain, in an iterative (and converging) manner. The optimization is achieved by applying the MCF algorithm on an augmented graph related to each domain. Simulation shows that our approach optimizes the substrate resource utilization compared to existing method. Besides, our method is easy to deploy in current Internet architecture.

References

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer*, no. 4, pp. 34–41, 2005.

[2] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, A. Wundsam, M. Kind, O. Maennel, and L. Mathy, "Network virtualization architecture: proposal and initial prototype," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pp. 63–72, ACM, 2009.

[3] A. Belbekkouche, M. Hasan, and A. Karmouch, "Resource discovery and allocation in network virtualization," *Communications Surveys & Tutorials, IEEE*, vol. 14, no. 4, pp. 1114–1128, 2012.

[4] A. Fischer, J. F. Botero, M. Till Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 4, pp. 1888–1906, 2013.

[5] Y. Zhu and M. H. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *INFOCOM*, vol. 1200, pp. 1–12, 2006.

[6] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, 2008.

[7] N. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *INFOCOM*, pp. 783–791, IEEE, 2009.

[8] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pp. 81–88, ACM, 2009.

[9] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 2, pp. 38–47, 2011.

[10] G. Sun, H. Yu, V. Anand, and L. Li, "A cost efficient framework and algorithm for embedding dynamic virtual network requests," *Future Generation Computer Systems*, vol. 29, no. 5, pp. 1265–1277, 2013.

[11] D. Dietrich, A. Rizk, and P. Papadimitriou, "Multi-domain virtual network embedding with limited information disclosure," in *IFIP Networking Conference, 2013*, pp. 1–9, IEEE, 2013.

[12] M. Shen, K. Xu, K. Yang, and H.-H. Chen, "Towards efficient virtual network embedding across multiple network domains," in *Quality of Service (IWQoS), 2014 IEEE 22nd International Symposium of*, pp. 61–70, IEEE, 2014.

[13] M. Chowdhury, F. Samuel, and R. Boutaba, "Polyvine: policy-based virtual network embedding across multiple domains," in *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, pp. 49–56, ACM, 2010.

[14] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Computer Networks*, vol. 55, no. 4, pp. 1011–1023, 2011.

[15] K. Guo, Y. Wang, X. Qiu, W. Li, and A. Xiao, "Particle swarm optimization based multi-domain virtual network embedding," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pp. 798–801, IEEE, 2015.

[16] C. Wang, S. Shanbhag, and T. Wolf, "Virtual network mapping with traffic matrices," in *Communications (ICC), 2012 IEEE International Conference on*, pp. 2717–2722, IEEE, 2012.

[17] sndlib *http://sndlib.zib.de*.

[18] GT-ITM *http://www.cc.gatech.edu/projects/gtitm/*.