An Approach Towards Resource Efficient Virtual Network Embedding

Adil Razzaq, Muhammad Siraj Rathore
School of ICT, KTH-Royal Institute of Technology
Stockholm, Sweden
{arazzaq, siraj}@kth.se

Abstract— Network virtualization is at the heart of efforts to end Internet ossification, and utilize network infrastructure efficiently. The key concept is to share the infrastructure resources among many users at the same time and in such a way to enable them to deploy the required architectures. This leads to virtual networks (VNs), demanding different resources that have to be embedded on the underlying shared infrastructure network. The requirements of a VN can be based on factors such as, the type of traffic it needs to carry. Some portion of the same resources (substrate resources) needs to be assigned to every VN and the substrate resources are also finite. Thus, a mechanism needs to be devised in order to schedule the resources. This embedding process with resource constraints on virtual nodes and links to be applied on the substrate, which also has limited resources, is challenging and corresponds to the category of NP-hard problems. In this paper, we propose an approach to solve this problem by mapping the vertices of the VN as closely as possible in the substrate network and then assigning virtual edges to the shortest paths which satisfy their demands. This could enable the substrate to accommodate more VNs in the same resource database and hence optimize the substrate's bandwidth utilization. To the best of our knowledge closest node mapping proposed in this solution is a novel approach and is evaluated and compared to the existing approach of greedy node mapping in different scenarios. The sensitivity analysis by varying the different parameters and their effect on mapping VNs is also presented.

Keywords- *Network Virtualization; Virtual Network Embedding; Resource Management; Bandwidth Optimization; Scalability.*

I. INTRODUCTION

Networks have a finite number of resources. Network virtualization enables the sharing and use of resources simultaneously by many users. It is an important proposal towards ending the Internet ossification [6, 8, 10]. A number of virtual networks (VNs) can be deployed on the physical or substrate network, depending on the capability of the substrate and demands of the VNs. The VNs can require different number and types of resources depending on many factors, e.g., a VN carrying delay sensitive or throughput sensitive traffic.

As proposed in [8], the future Internet architecture could be divided into Infrastructure Providers (InfPs) and Service Providers (SrvPs). The InfPs can manage physical infrastructure whereas the SrvPs can deploy network

protocols and offer end to end services. This can enable end to end provision of many important services, e.g., Quality of Service (QoS). However to enable this the InfPs must be able to keep track of their available resources and those assigned to SrvPs. Likewise the SrvPs should also have a record of what resources they received from InfPs and allocation made to their clients.

To make it short, when a VN request is received, a decision needs to be made whether it can be satisfied based on the available network resources or not? In other words, we need to embed or map a VN with resource constraints onto the substrate network (SN), which has limited resources. The assigning of resource constrained VNs to a substrate with finite resources is a challenge and falls in the class of NP-hard problems.

Various solutions have been derived in the past for embedding VNs onto physical networks. By considering that the substrate resources are finite, we make “admission control” an integral part of our solution in contrast to the previous solutions proposed in [3, 5, 11]. The proposed solution considers that VN requests are dynamic and are received online which is contrary to [5, 11]. Some previous work points to specific virtual topologies [11], whereas our solution is not limited to any specific topology. Our approach is inspired to some extent by [7] as we use some notations to denote both virtual and substrate networks. However, the solution explained in [9] conceives support for path splitting whereas [7] also considers path migration from the physical network. Path splitting & migration incur cost complexities and are not in the scope of the proposed solution.

The approach of mapping virtual vertices to the substrate nodes with most resources was presented in [7], although this approach can utilize node resources effectively but can make the nodes placed far apart from each other and can result in relatively longer paths. As a consequence, in the edge mapping phase resources can be reduced from longer substrate paths. The proposed solution focuses on placing the virtual nodes as closely in the substrate as possible. This ensures that the paths found for the edge mapping are the shortest in length. The virtual vertex is mapped on the first physical node that satisfies its demand. The remaining vertices are mapped in the same manner. Since a single path may not be able to satisfy the resource constraints so, k-shortest paths algorithm [2] has been used for path finding between each pair of nodes. This enables the mapping of the

edge on the second, third or the k^{th} path if the initial ones are not able to satisfy its demand.

The main contributions of this paper are: The impact of varying k for different substrates and VNs has been analyzed. The paper also evaluates various substrate networks to analyze which type of network can map more number of VNs. A comparison with the greedy node mapping [7] with the approach presented in the proposed solution (closest node mapping) is also given. The proposed solution can map both the random as well as topology specific VNs based on standard network topologies (star, bus, ring, mesh). The proposed solution starts with the hypothesis that a well connected network with an optimal value of k , will be able to map more VNs. A well connected network is defined as the one which has more links as well as resources on them. The optimal value of k for both the proposed as well as compared solution has been calculated and presented here.

The rest of the paper is organized as follows. Section II defines the problem while Section III proposes the solution. Section IV presents simulation results. Section V specifies the future work and winds up the discussion with authors' conclusions.

II. NETWORK MODEL & PROBLEM DESCRIPTION

In the VN embedding problem, the substrate or physical network is represented in the form of a graph. The VN requests are also received in the form of a graph. In the proposed solution both the virtual as well as substrate networks are represented as undirected graphs. Physical network is represented by, $P = (N, N_C, L, L_C)$. Where P represents a physical network which has nodes (N), node capacity (N_C), links (L) and capacity of links (L_C).

Virtual network is represented by, $A = (V, V_C, E, E_C)$. Where A denotes a network (VN) which needs to be assigned to the substrate and it has vertices or nodes (V), vertex capacity (V_C), edges (E) and E_C denotes required capacity for the edges.

Throughout this document when a reference is made to a link or node, it means that it belongs to the substrate, while VN's link & node will be termed as edge and vertex respectively. For nodes and vertices Central Processing Unit (C.P.U.) whereas, for edges and links, bandwidth is considered as a resource.

Fig. 1 shows a substrate network whereas Fig. 2 represents a VN request. The notation for describing the node and link capacities is similar to the one proposed in [7].

The VN will only be mapped on the substrate if the requirements of each of its vertex as well as edge are met. After mapping vertices onto the nodes which satisfy vertex demand, paths need to be calculated for each pair of nodes in the VN. Then the link resources in the path are compared with the edge demand. At this point if the path satisfies the edge request then the VN is completely mapped.

The proposed solution can map both the topology specific VNs (e.g., VN represented in Fig. 2 is based on mesh topology) as well as randomly generated VNs.

Figure 1: Substrate Network

Figure 2: Virtual Network

III. VN ASSIGNMENT

When a VN request is received, substrate's node resources need to be assigned to its vertices and link resources to the edges. In the proposed solution, vertices are assigned to nodes, then edge disjoint k -shortest paths [2] are found between each pair of nodes and in the last step edges are mapped on the links that satisfy the edge demand.

In order to match VN demands with substrate resources we need to define a mapping function. We have, $A = (V, V_C, E, E_C)$, which has to be mapped to, $P = (N, N_C, L, L_C)$.

In the first step of our approach, we map the vertex requiring most resources to the node satisfying its demand. So, our node mapping function is:

$$F(N) = \text{Max}(V, V_C) \rightarrow N, N_C \\ \text{Iff } V_C \leq N_C \quad (1)$$

The process of mapping the vertex which demands most resources first, ensures that if the substrate cannot satisfy its demand then the process stops here for this VN and we don't need to check the demands of the remaining vertices. If the demand of the first vertex is satisfied then the process is repeated for all the remaining vertices moving on from vertex demanding most to least resources and once all of them are assigned to nodes the solution moves to the edge mapping phase.

So, as a second step we need to map an edge (E) on substrate path (S_p). In other words for each edge we need to find a substrate path which satisfies the edge demand, i.e., each link of the path should be capable of fulfilling the edge

request. In the proposed solution, first k-shortest paths are found for each edge. We then take the minimum of all the links in each path, i.e., the link with the minimum resources in the path. After doing so the edge demand is matched with it, which is done to make sure that the link with the least resources in the path is able to match the edge demand. If it can, then the remaining links will certainly be able to do that. The mapping function $F(E)$ for edge mapping is [7,9]:

$$\begin{aligned} P &= \text{Min} (S_P (\sum_{i=1:n} L_i)) \\ F(E) &= P \quad \forall E \in A \\ \text{Iff } E_C &\leq L_C \end{aligned} \quad (2)$$

where $F(E)$ defines the mapping function for the edge, which is to be mapped on the first substrate path (S_P) having 1 or more than one links ($\sum_{i=1:n} L_i$) with sufficient resources to satisfy edge demand; we need to do this for all the edges ($\forall E$) of the virtual network (A). The edge is only mapped on the path if the value returned by the mapping function is greater than or equal to the edge demand ($E_C \leq L_C$).

By using the k-shortest paths algorithm we find more than one paths for each pair of vertices. We then calculate resources on the k shortest paths (k can be given any value 2,3,4,5 etc.) and map the edge on the first available path which has sufficient resources (shortest path mapping (SPM), algorithm). This ensures that the resources are only assigned from the paths which are shortest among the k paths [7]. The proposed as well as compared solutions have been evaluated for different values of k and give interesting results for different VNs and network environments.

We now describe the two main algorithms of our solution starting from closest node mapping (CNM):

Algorithm 1: Closest node mapping

Step 1: Take the first request at any instant of time.
Step 2: Find the subset S of substrate nodes having sufficient resources that satisfy the CPU demands of the VN.
Step 3: If VN request is satisfied at this stage then call “shortest path mapping algorithm”, otherwise go to step 4.
Step 4: Store the request at the end of the request queue.
Step 5: Go to step 1.

We satisfy VN requests by using the first come first served (FCFS) approach and begin by assigning their vertices to the substrate nodes. The path mapping algorithm is called only if the node request can be satisfied at this stage. Otherwise the request is stored at the end of the queue to give it a chance to be mapped on the substrate in future.

Once the nodes are mapped then we need to find paths with sufficient resources for each pair of nodes which satisfy the demands of VN's edges. So, now we formulate the path mapping algorithm.

Algorithm 2: Shortest path mapping

Step 1: Take the request which has successfully passed the node mapping stage.
Step 2: Search the k-shortest paths for each edge.
Step 3: Calculate the number of resources on each path.
Step 4: Select the first path which satisfies the edge demand.
Step 5: If request is satisfied then call “closest node mapping algorithm”. Otherwise go to step 6.
Step 6: Store the VN at the end of the request queue and go to step 7.
Step 7: Call “closest node mapping algorithm”.

The path mapping phase of the solution starts by finding the edge disjoint k-shortest paths for each pair of nodes of the VN. In the next step the resources on each of these paths are calculated and the shortest path having adequate resources satisfying the edge demand is selected. Otherwise edge request is mapped with the 2nd, 3rd upto kth path. At this point either the VN is completely mapped on the substrate (request satisfied) or is placed at the end of the request queue (request not satisfied).

IV. EXPERIMENTAL SETUP & EVALUATION

This section starts with the description of the experimental setup, after that the evaluation results are presented. The main focus of evaluation is to see the benefit of closest node mapping and its effect on mapping different sets of VNs on various substrates by varying the value of k (for k-shortest edge disjoint paths). A comparison with the greedy node mapping approach [7] and impact of increasing the value of k on it is also presented. The proposed solution has been implemented using Matlab®.

A. Experimental Setup

The substrate networks have been generated using Matlab® as well as the BRITE tool [1] whereas the virtual networks have been created using Matlab®.

1) Substrate Networks

The proposed solution has been tested on different substrate networks. Each of these networks consists of 100 nodes and the node resource (C.P.U.) as well as link resource (bandwidth) are distributed from 1 to 100 units [4,7]. The main difference among the substrate networks generated using Matlab® is in the manner in which their nodes are connected together with different probabilities (0.5, 0.4). The substrate networks generated using the BRITE tool differ among each other in the number of links (500, 400, 300). The reason for generating different substrates from different tools is to evaluate the effect of VN mapping on various types of networks.

2) Virtual Networks

Four different sets of VNs have been mapped onto the substrate networks, each set differs from the other, either by number of vertices or edges.

Set 1: The number of vertices of a VN is randomly chosen between 2 and 10 and the vertices are randomly connected with the probability 0.3.

Set 2: The number of vertices of a VN is randomly chosen between 2 and 10 and the vertices are randomly connected with the probability 0.5.

Set 3: The number of vertices of a VN is randomly chosen between 2 and 20 and the vertices are randomly connected with the probability 0.5.

Set 4: The number of vertices of a VN is chosen between 3 and 10. These are topology specific VNs based on standard network topologies (star, bus, ring and mesh). The number of edges of a VN depends on its topology.

The set-2 is similar to the setup presented in [4, 7, 11] while set-3 resembles with the one given in [7].

The odd-numbered Figs.3, 5 and 7 below represent mapping VNs on the substrate generated using BRITE containing 500 links, while even-numbered Figs.4, 6 and 8 show the same on the substrate having 400 links. Results for 300 links substrate are only shown in table 1 (due to space limitations, an exception is Fig. 9 which is presented to support observation no.5) and results generated on substrates obtained using Matlab® are also omitted due to the same reason. The results indicate mapping VNs up to a point where substrate network's resources are almost exhausted.

B. Evaluation

Since, placing nodes near to each other so that the paths found should be shortest is an integral part of the proposed solution. So, it has been tested and compared in different network environments and for various values of k (2, 3, 4, 5, 6) to find an optimal value of k for better VN mappings in a given scenario. The evaluations started with the lowest value of k ($k=2$) and stopped at a point where further increase did not produce a positive effect on embedding VNs onto the substrate. A VN is mapped only if requirements of both its vertices as well as edges are satisfied. Important observations related to the simulation results are presented below.

- (1) Benefits of increasing the value of k .
 - (i) Gives us more choice to map an edge on a path between a pair of nodes.
 - (ii) The increased choice for path selection (up to an optimal value) results in more VNs being mapped on a substrate.
- (2) Drawbacks of increasing the value of k .
 - (i) Smaller values of k give shortest paths. As we increase value of k the paths will still be shortest but comparatively longer than the paths with lower values of k e.g., path found with $k=1$ will be the shortest of all as we increase k we will get comparatively longer paths.
 - (ii) Edges are mapped on longer paths and as a consequence, resources could be reduced from more links after mapping.
- (3) Higher (but optimal) values of k (3,4,5) result in more VNs being mapped on the substrate (Figs.3-8), which

verifies second part of the hypothesis, “an optimal value of k , will enable more VNs being mapped on a substrate”.

Figure 3: Random VNs, Vertices: 2-10, Edge_Prob: 0.3

- (4) In most cases (Figs.3-8) $k=5$ results in better VN mappings than the higher value of k ($k=6$). So, we cannot keep on increasing k and it should be limited to a reasonable value.

Figure 4: Random VNs, Vertices: 2-10, Edge_Prob: 0.3

- (5) The lowest value of k ($k=2$), in most evaluations produces less VN mappings (Figs.3-8) than the higher values of k . This happens because we have very limited number of alternatives (only 1) in terms of paths to choose from if the first one does not satisfy the edge demand.

However, in some situations, in case of a small substrate (e.g., substrate containing 300 links) $k=2$ produces more VN mappings than the highest evaluated value of k ($k=6$) as shown in Fig. 9, which depicts mapping one set of VNs (set 2), on the said substrate. This happens because when we set a very high value of k for a comparatively small substrate then its most links get exhausted for VNs arriving earlier and very few resources are left for VNs arriving later.

- (6) The well connected substrate (Figs. 3, 5, 7) is capable of mapping more VNs than the less connected one (Figs. 4, 6, 8) which proves first part of the hypothesis, “a well connected network will be able to map more VNs”.

(7) More number of less connected VNs (link probability = 0.3) can be mapped (Figs. 3 and 4) on the same substrate as compared to more connected VNs (link probability = 0.5) as shown in Figs. 5 and 6. This is because less connected VNs demand less link resources and therefore the substrate can accommodate more VNs of this type.

Figure 5: Random VNs, Vertices: 2-10, Edge_Prob: 0.5

Figure 6: Random VNs, Vertices: 2-10, Edge_Prob: 0.5

(8) VNs with the same probability but less number of vertices (2-10) can be embedded more (Figs. 5 and 6) than the VNs with more number of vertices (Figs. 7 and 8). Higher number of vertices in a VN demands more connections between them and the substrate can only map the VN if it can satisfy the requests of all the edges.

(9) The order of VN arrival also affects its mapping (not shown in figures due to space limitations) and the successful mapping of a VN depends both on its demands as well as the state of the substrate at the time of its arrival.

(10) Different types of substrate networks can give different results in terms of mapping number of VNs. To explore this we created 2 different substrate networks using Matlab® and mapped the above mentioned sets of VNs on them (as

Figure7: Random VNs, Vertices: 2-20, Edge_Prob: 0.5

Figure 8: Random VNs, Vertices: 2-20, Edge_Prob: 0.5

Figure 9: Random VNs, Vertices: 2-10, Edge_Prob: 0.5

discussed in experimental setup section). The results from these substrates confirm above mentioned observations 1, 2, 5,6,7,8 & 9. In this case observations 3 & 4 do not hold in the same way as we always get more mappings for highest value of k (k=6) due to more number of links in these substrates. However these substrates map less number of VNs as compared to above mentioned substrates as although

they are comprised of more links yet, they contain fewer resources on them.

The effect of increasing the value of k on VN mapping using the greedy node mapping approach [7] on substrates created using BRITE can be summarized in Table 1. The table shows mapping results of one set of VNs (set 2).

TABLE I. EFFECT OF K & % OF MAPPED VNS (GNM-SPM)

K	300	400	500
2	37 %	43 %	38 %
3	40 %	48 %	43 %
4	36 %	47 %	51 %
5	34 %	41 %	48 %
6	32 %	40 %	45 %

Important observations include:

- I. The results confirm the above mentioned observations 1 and 2.
- II. The lower values of k are optimal in this case (2, 3, 4) for smaller networks (300 and 400) and higher values (4, 5, 6) for larger network (500) so observations 3 and 4 change here.
- III. Observation 5 does not hold here for smaller networks (300 and 400).
- IV. Observations 6, 7, 8 and 9 also hold for this solution.
- V. A comparison with Figs. 5 and 6 and with smaller substrate (Fig. 9) suggests that best case of GNM has mapped 2% more VNs for smaller substrate (300), whereas CNM does the same for a larger network (500). This can be analyzed by comparing ($k=4$) in the 500 links column of Table 1 with $k=5$ in Fig. 5.

V. CONCLUSION AND FUTURE WORK

Virtual network embedding is a core issue in the area of network virtualization. In the past this problem has been solved by using different approaches. It is an NP-hard problem; having many constraints and providing an optimal solution is a challenge. Through this paper we have proposed a technique to utilize the substrate resources efficiently and effectively. This better utilization of the substrate may be achieved at the cost of using more computational resources and also it may take relatively more time to embed a VN using this approach. Nevertheless it could allow the substrate to embed more VNs on same physical resources. The results have verified the hypothesis, and a more connected network with an optimal value of k , maps more number of VNs both for the proposed as well as the compared solution.

The baseline VN embedding algorithm [7] performs batch processing of VN requests where it sorts VNs based on their revenues and satisfies the request with the most resources first. Currently we are implementing this approach of the said algorithm (embedding VNs based on revenues) to evaluate its effect on VN mapping against the first come first served approach presented in the proposed solution. Since one of the findings of this paper is that "order of VN arrival also affects its mapping", so first satisfying VN

which demands most resources should give interesting results.

ACKNOWLEDGMENTS

We would like to thank Suleman Khan (KTH University, Sweden), Luca Giraudo (Politecnico Di Torino, Italy) and Minlan Yu (Princeton University, U.S.A.) for their guidance during this work.

REFERENCES

- [1] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: Universal topology generation from a user's perspective", Boston University, BUCS-TR- 2001-003, 2001.
- [2] D. Eppstein, "Finding the k shortest paths", In Proc. IEEE Symposium on Foundations of Computer Science, pp. 154-165, 1994.
- [3] J. Fan and M. Ammar, "Dynamic topology configuration in service overlay networks: A study of reconfiguration policies", In Proc. IEEE INFOCOM, 2006.
- [4] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection", ACM SIGCOMM, 2009.
- [5] J. Lu and J. Turner, "Efficient mapping of virtual networks onto a shared substrate", Washington University, Technical Report WUCSE-2006-35, 2006.
- [6] J. S. Turner and D. E. Taylor, "Diversifying the Internet", In Proc. IEEE GLOBECOM, pp. 755-760, 2005.
- [7] 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.
- [8] N. Feamster, L. Gao, and J. Rexford, "How to lease the Internet in your spare time", *SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 61-64, 2007.
- [9] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping", IEEE INFOCOM, pp. 783-791, 2009.
- [10] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization", IEEE Computer Magazine, vol. 38, no. 4, pp. 34-41, 2005.
- [11] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components", In Proc. IEEE INFOCOM, pp. 1-12, 2006.