

# Flexible Substrate Network to Support Virtual Network Embedding

Minlan Yu, Yung Yi, Jennifer Rexford, Mung Chiang  
Princeton University

## 1 Virtual Network Embedding

Network virtualization provides a powerful way to run multiple networks, each customized to a specific purpose, at the same time over a shared substrate [1]. Recent proposals for virtual network infrastructure try to build a diversified Internet to support a variety of network services and experiments through a shared substrate [2, 3]. Network virtualization also plays an important role to support multiple architectures simultaneously as a long-term solution for the future Internet [4].

The network virtualization projects on both experiments and architectures all grapple with the technical challenges of determining how to deploy virtual networks for the efficient utilization of the substrate network.

In practice, the substrate network usually keeps providing services for the upcoming requests. So here we consider the online virtual network embedding problem. We view the substrate network as an undirected graph  $G_s = (V_s, E_s)$ . At time  $T_i$ , the  $i$ th virtual network request comes. We also use an undirected graph  $G_v^i = (V_v^i, E_v^i)$  to denote this virtual network. We need to find a mapping from each virtual network to the nodes and links in the substrate, or postpone this request if we don't have enough resources. We describe this mapping with two steps: (i) **Node Mapping**  $f_N^i : V_v^i \rightarrow V_s$ ; (ii) **Link Mapping**  $f_L^i : E_v^i \rightarrow P_s$  where  $P_s$  is a group of paths in  $G_s$ . Some requests may have special requirements of node capacity (eg: CPU resources) and link capacity (eg: bandwidth). The substrate network has limited CPU and bandwidth resources, leading to the question of how to allocate these resources to the virtual network and meet their requirements.

Previous papers [5, 6, 7] work on different aspects of this problem, and propose heuristic algorithms to find suboptimal solutions. Their work shows that even an aspect of virtual network embedding problem is already extremely hard to find an optimal solution.

This led us to rethink the virtual network embedding problem. Rather than work on the difficult problems, we can make the substrate network more friendly and supportive to virtual network embedding. We find out that if we make the substrate network more flexible, we can make the problems easier and the solutions more efficient.

This flexibility includes two parts: allowing substrate path splitting and allowing migration. We will explain the two parts in the section 2 and 3.

## 2 Allowing Substrate Path Splitting

For one virtual link, it's natural to map it to one path in the substrate. But forcing each virtual link to map to just one underlying path in the substrate often induces NP-hard embedding problems. If the substrate supports mapping a virtual link to multiple physical links/paths, this would essentially relax that constraint. From the user's perspective, he still has one virtual link. But in the substrate, we actually split the packets on the virtual link into a group of substrate paths. Allowing flexible path splitting in the substrate can makes it possible to solve the link-embedding problem in polynomial time. Also, with this flexibility, we can get better utilization of the substrate resources than not allowing path splitting.

In some sense, this substrate path splitting is similar to multi-path routing. It can also be used to

ease congestion and overcome the failure of nodes and links. Today, splitting traffic over multiple path with equal cost is common. This is also a good evidence showing that substrate path splitting can be allowed and is even desirable.

Some people would be concerned that if they allow path splitting in the substrate, the link would behave differently than they might expect. For example, there may be variable propagation delay on different substrate paths for one virtual link. The packets sent along the single virtual link may arrive out of order. To solve these problems, we can add artificial delay to equalize the delay on different paths. We can also use hash-based splitting as multi-path routing does to reduce the effects of out-of-order packets. Our substrate path splitting approach will not influence the behaviors on the virtual network much, when it achieves better substrate resource utilization.

### 3 Migration

The online embedding problem would be much easier to solve if we could migrate an already-running virtual network to make room for the new coming ones. In this way, we may attain more efficient utilization of the substrate resources by accepting more requests. Migration may also happen when failure occurs. If some substrate links or nodes fail, we have to move the virtual network on these nodes or links to another place. The third scenario that need migration is the dynamic requirement of traffic matrix. We may have to change the embedding of a virtual network to meet with its new traffic pattern or to achieve less cost in the substrate network.

In general, there are three migration ways the substrate network can provide: (i) tune the splitting ratio (only for those virtual network requests that allow path splitting); (ii) migrate the routing path; (iii) migrate the nodes.

Again people may worry about the efficiency of migration. They may also argue that to do the migration (node migration in particular), we have to stop the ongoing operations in virtual networks. This actually is not a big problem. We will show that all the three ways of migration are doable. They will not in-

terrupt the operations in virtual networks much and don't cost much time or effort in practice.

For (i) we only need to change the parameter of splitting ratio in the routers. (ii) Migrating routes isn't all that disruptive as it is much like path splitting. We tend to migrate routes rather than nodes in our algorithm if possible.

Sometimes, migrating nodes is also necessary for better resource utilization. The node migration is also operable because (1) long-running services may have maintenance windows of their own, where they drain traffic off a server to upgrade the software, so the virtual node could be migrated at this time. (2) in general, with ample warning, the migration could be done at a time that it is less disruptive, and with prior planning.

In addition, migration may not consume much time even for node migration. [8] shows that they are able to migrate virtual machine in a few seconds, in order to eliminate hot spots in data centers. We can use similar technology here.

### References

- [1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, 2005.
- [2] "GENI." <http://www.geni.net/>.
- [3] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI veritas: Realistic and controlled network experimentation," in *Proceedings of ACM SIGCOMM 2006*, (Pisa, Italy), September 2006.
- [4] N. Feamster, L. Gao, and J. Rexford, "How to lease the internet in your spare time," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 61–64, 2007.
- [5] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. IEEE INFOCOM*, 2006.

- [6] J. Fan and M. Ammar, "Dynamic topology configuration in service overlay networks: A study of reconfiguration policies," in *Proc. IEEE INFOCOM*, 2006.
- [7] J. Lu and J. Turner, "Efficient mapping of virtual networks onto a shared substrate," 2006. Technical Report, Washington University.
- [8] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in *Proc. Networked Systems Design and Implementation*, (Cambridge, MA), April 2007.