Prototype Development of a Virtual Network Embedding System Using OpenStack

Itsuho Goda[1], Kohei Sato[1], Yukinobu Fukushima[1], Heung-Gyoon Ryu[2] and Tokumi Yokohira[1]
[1] The Graduate School of Natural Science and Technology, Okayama University
Tsushima-naka 3-1-1, Kita-ku, Okayama-city, Okayama 700-8530, Japan
[2] Department of Electronics Engineering, Chungbuk National University
Cheongju, Korea 361-763
E-mail: fukusima@okayama-u.ac.jp, yokohira@okayama-u.ac.jp

Abstract: Network virtualization enables us to make efficient use of resources in a physical network by embedding multiple virtual networks on the physical network. In this paper, we develop a prototype of a virtual network embedding system. Our system consists of OpenStack, which is an open source cloud service platform, and shell scripts. The shell scripts automatically construct the required virtual network on the physical network using OpenStack command-line interface. Experimental evaluation confirms that our system constructs a requested virtual network and assigns computing and networking resources to it appropriately.

Keywords—**Virtual network embedding, OpenStack**

1. Introduction

Network virtualization has attracted increasing attention in both industry and academia [1, 2]. In network virtualization, multiple virtual networks consisting of virtual nodes and virtual links are embedded on a single physical network consisting of physical nodes and physical links. As a result, we can share and effectively use the resources in the physical network among multiple virtual networks.

In embedding a virtual network on a physical network, we have to determine to which physical nodes/routes virtual nodes/links are mapped so that a predetermined objective function is optimized under some constraints (e.g., resource constraint). This problem is called virtual network embedding (VNE) problem [3].

A large number of algorithms (VNE algorithms) for VNE problem have been proposed in the literature [3-8]. However, they are evaluated only in computer simulations. In order to evaluate various VNE algorithms in a real situation with various performance measures (e.g., virtual network provisioning time), we need a VNE system, which actually constructs virtual networks on a physical network.

In this paper, we develop a prototype of a VNE system. Our system uses OpenStack [9] as a base platform. OpenStack is an open source cloud service platform for Infrastructure-as-a-Service (IaaS), and enables us to generate a tenant network, which consists of virtual machines (VMs) and virtual links, in data centers. We can use the tenant networks generated by OpenStack as virtual networks in network virtualization by implementing network functions (e.g., router) on the VMs in the tenant networks and regarding them as virtual nodes. Our system consists of OpenStack and shell scripts. The shell scripts automatically construct the required virtual network on the physical network using OpenStack command-line interface. We experimentally evaluate our system.

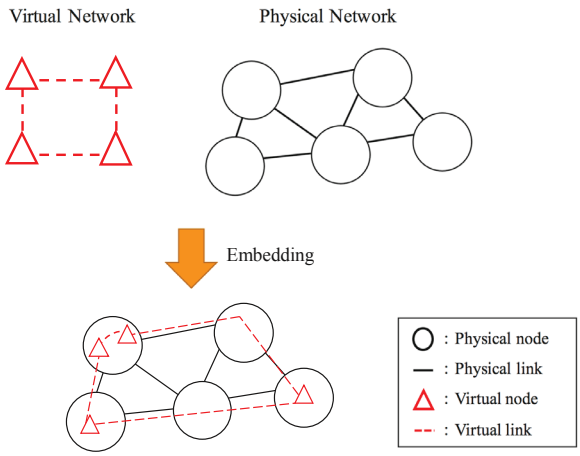

Figure 1. Example of virtual network embedding.

2. Virtual Network Embedding and OpenStack

2. 1 Virtual Network Embedding

In VNE problem, we have to determine how the requested virtual network is mapped to the physical network so that some objective function (e.g., to maximize economical profit of the physical network provider) is optimized (Fig. 1). VNE problem is divided into two subproblems: *node mapping problem* and *link mapping problem*. In node mapping problem, we have to determine to which physical node each virtual node is mapped. In link mapping problem, we have to determine to which physical route each virtual link is mapped.

In node/link mapping problems, resource constraint has to be satisfied. Each virtual node requests some node resources (CPU, memory, and storage), and it can only be mapped to physical nodes that have enough node resources. Similarly, each virtual link requests some link resources (bandwidth), and it can only be mapped to physical routes that have enough link resources.

2. 2 OpenStack

OpenStack [9] is an open source cloud computing platform for IaaS. With OpenStack, we can construct a tenant network consisting of VMs and virtual links in data centers.

OpenStack consists of multiple components. Among them, *Nova* and *Neutron* play an important role in our VNE system. Nova maintains node resources of physical computers and is responsible for generating and removing VMs, which correspond to virtual nodes in VNE. Neutron is responsible for generating and removing VLANs, which

correspond to virtual links in VNE. The operations of Nova and Neutron can be requested via WebGUI and command-line interface.

Neutron is not equipped with QoS control function in default setting. Therefore, we have to reserve link resources (i.e., bandwidth) for virtual links without using Neutron. We can realize QoS control by directly configuring Open vSwitch, which is a virtual switch used in OpenStack.

3. Virtual Network Embedding System

Figure 2 shows the procedure for VNE. In step 1, we determine node and link mappings using a VNE algorithm. In step 2, given the mapping result (VNE information), we actually construct the requested virtual network on the physical network. Our VNE system developed in this paper copes with only step 2.

In our VNE system, we assume that VNE information is described with BRITE format [10]. Figure 3 depicts an example of VNE information. The first line expresses the numbers of virtual nodes and links in the requested virtual network. The fifth to eighth lines show node mapping information. Each of the lines corresponds to mapping of a virtual node to a physical node. Each line consists of three informations: 1) ID of the virtual node, 2) node resource request ID (i.e., which operating system and how much CPU/memory/storage resources are requested by the virtual node), and 3) ID of the physical node to which the virtual node is mapped. The eleventh to fourteenth lines show link mapping information. Each of the lines corresponds to mapping of a virtual link to a physical route. Each line consists of four informations: 1) ID of the virtual link, 2) ID of the first virtual node connected by the virtual link, 3) ID of the second virtual node connected by the virtual link, and 4) link resource (bandwidth [kbps]) allocated to the virtual link. In this paper, we assume that virtual links are mapped to physical routes that OpenStack automatically determines.

Figure 4 depicts the constitution of our system. Our system consists of OpenStack and three programs or shell scripts: Preprocess.o, MakeVN.sh, and RsvBW.sh. The preprocess.o is a program written in C language while The MakeVN.sh and the RsvBW.sh are bash shell scripts. The detailed explanation of them are as follows.

The Preprocess.o does preprocessing. Given the VNE information, it generates and executes the MakeVN.sh.

The MakeVN.sh generates the VMs (i.e., virtual nodes) and reserves node resources for them using OpenStack command-line interface “nova boot”. It also generates the VLANs (i.e., virtual links) using OpenStack command-line interfaces “neutron net-create” and “neutron subnet-create”. It also generates and executes the RsvBW.sh.

The RsvBW.sh reserves bandwidths for the virtual links. In tenant networks in OpenStack, VMs are connected to VLANs via Open vSwitch, which is a virtual switch. The RsvBW.sh limits the inbound traffic rate of an ingress port of the Open vSwitch that the virtual node is connected to, so that the traffic transmission rate on the virtual link is kept below the allocated bandwidth.

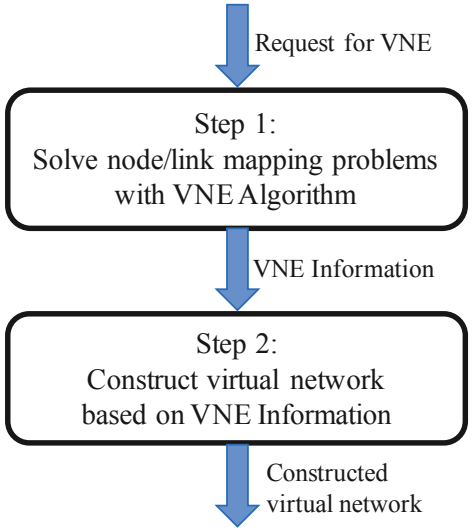

Figure 2. Procedure for VNE.

```
1   Topology: (4, 4)
2   Model
3
4   Nodes: (4)
5   0 1 1
6   1 1 2
7   2 1 3
8   3 1 1
9
10  Links: (4)
11  0 0 1 100000
12  1 0 3 10000
13  2 1 2 1000
14  3 2 3 100
```

Figure 3. VNE information.

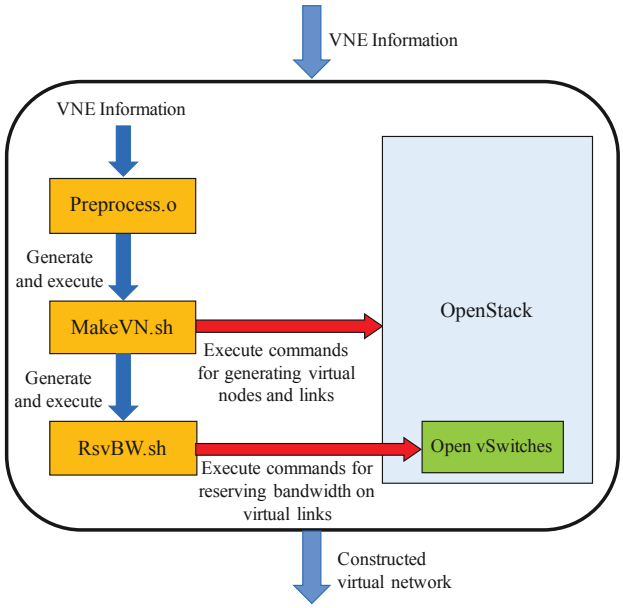

Figure 4. System constitution.

4. Experimental Evaluation

4. 1 Experimental Environment

Figure 5 depicts our experimental physical network and requested virtual network. The physical network is a star network consisting of four physical nodes and one physical switch. Physical node 0 is a controller of OpenStack while physical nodes 1 to 3 are used to accommodate virtual nodes and links. The virtual network is a ring network consisting of four virtual nodes and four virtual links. The resource requests of the virtual nodes and links are summarized in Table 1. We assume that a VNE algorithm decided that virtual nodes VN_0 and VN_3 are assigned to physical node 1, virtual node VN_1 is assigned to physical node 2, and virtual node VN_2 is assigned to physical node 3. We use 1) CentOS 7.1 for OSes of both physical and virtual nodes and 2) OpenStack version Liverty in our system.

4. 2 Results

Figure 6 shows the virtual network constructed by our system, which is outputted by the OpenStack. In the figure, icons of display correspond to virtual nodes and icons of cloud correspond to virtual links. We can see that the requested four-node ring virtual network is appropriately constructed on our system.

We next check whether node resources are appropriately assigned to virtual nodes. We logged in each of the virtual nodes and confirmed that all of the four virtual nodes are equipped with the requested CPU/memory/storage resources shown in Table 1.

We finally check reachability between virtual nodes via a virtual link and whether link resources are appropriately assigned to virtual links. We confirmed reachability between any two virtual nodes that are directly connected via a virtual link using the ping command. In order to check whether requested bandwidth are assigned to each virtual link, we measured throughputs of them using iPerf. Figure 7 shows the results. In the figure, we can see that the throughput of each virtual link is bounded by its requested bandwidth shown in Table 1.

5. Conclusions

In this paper, we have developed a prototype of a virtual network embedding system using OpenStack. Experimental evaluation confirms that our system correctly constructs a required virtual network on a physical network.

One of our future work is to extend our system so that it can specify a physical route of a virtual link.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number 15K00129.

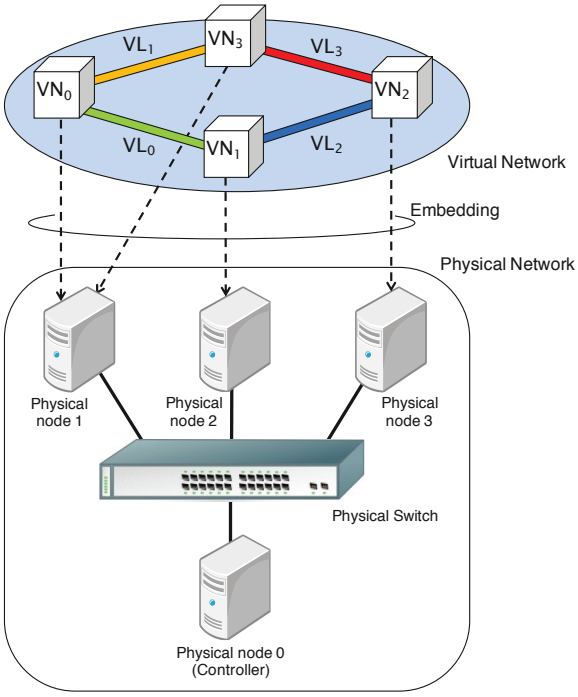

Figure 5. Experimental physical network and requested virtual network.

Table 1. Resources requested by virtual nodes and links.

Virtual nodes or links		Requested resources
Virtual nodes	VN_0	1 CPU, 1GB memory, 3GB storage
	VN_1	1 CPU, 1GB memory, 3GB storage
	VN_2	1 CPU, 1GB memory, 3GB storage
	VN_3	1 CPU, 1GB memory, 3GB storage
Virtual links	VL_0	Bandwidth of 100,000 Kbps
	VL_1	Bandwidth of 10,000 Kbps
	VL_2	Bandwidth of 1,000 Kbps
	VL_3	Bandwidth of 100 Kbps

References

[1] K. Tutschku, T. Zinner, A. Nakao and P. Tran-Gia, "Network Virtualization: Implementation Steps towards the Future Internet," in Proc. of Workshop on Overlay and Network Virtualization, May 2009.

[2] N. M. M. K. Chowdhury, "Network Virtualization: State of the Art and Research Challenges," IEEE Communications Magazine, vol. 47, no. 7, pp. 20–26, Jul. 2009.

[3] A. Fischer, J. F. Botero, M. T. Beck, H. Meer, and X. Hesselback, "Virtual Network Embedding: A Survey," IEEE Communications surveys & Tutorial, Vol. 15, Iss., 4, pp. 1888-1906, Feb. 2013.

[4] 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, Iss. 2, Apr. 2008.

[5] N. M. M. K. Chowdhury, M. R. Rahman and R. Boutaba, "Virtual Network Embedding with Coordinated Node and Link Mapping," in Proc. of INFOCOM, pp. 783-791, Apr. 2009.

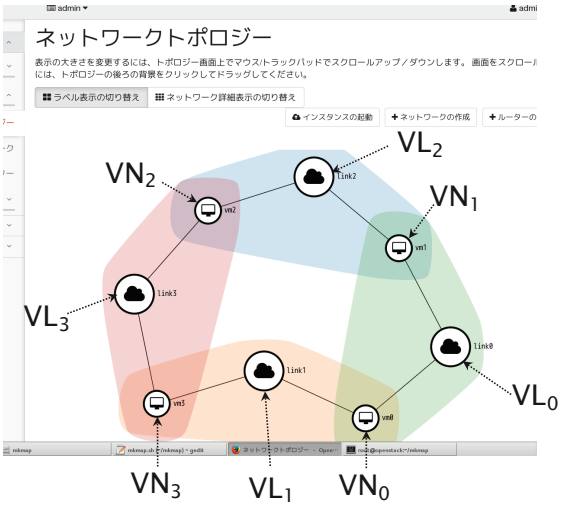

Figure 6. Virtual network constructed by our system.

[6] 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, Apr. 2011.

[7] A. Fischer, M. T. Beck and H. D. Meer, "An approach to Energy-efficient Virtual Network Embeddings," in Proc. of IM, pp. 1142–1147, May 2013.

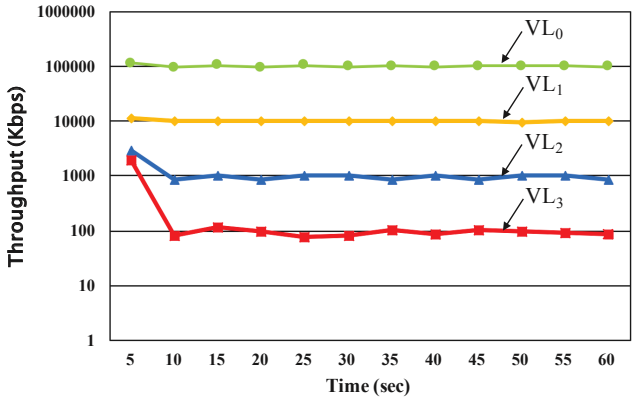

Figure 7. Throughputs of virtual links.

[8] H. Sugiyama, Y. Fukushima and T. Yokohira, "Performance Evaluation of an Energy Efficient Virtual Network Mapping Method - In the case of load-depending power consumption model -," in Proc. of ICESS, 11 pages, July 2015.

[9] OpenStack project website. http://www.openstack.org

[10] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: Universal Topology Generation from a User's Perspective," Technical Report, 47 pages, Boston University, 2001.