Send Orders for Reprints to reprints@benthamscience.ae

The Open Cybernetics & Systemics Journal, **2015**, *9*, 1449-1452

Open Access

Research and Implementation on Virtual Network Mapping Mechanism based on SDN

Yanqin Mao*, Yanxing Gui and Subin Shen

School of Computer Science & Technology, School of Software, Nanjing University of Posts and Telecommunications, Nanjing, 210003, P.R. China

Abstract: Network virtualization technology allows multiple logical networks or network applications to run simultaneously on the underlying physical network. The virtual network mapping is the key method to realize network virtualization. Virtual tenant network (VTN) is a new technique for construction of multi-tenant network in software definition networking (SDN) network. VTN mapping mechanism is a core problem. In this paper MAC address-based mapping method which maps the host to the virtual network according to the host's MAC address is proposed. Besides, a virtual network based on MAC address mapping method is established to verify the effectiveness of the VTN mapping method based on MAC address.

Keywords: Network virtualization, software defined networking, virtual network mapping, virtual tenant network.

1. INTRODUCTION

SDN [1] is a new network architecture. The main difference between SDN network and traditional network is that the control function of network equipment in SDN is separated from data forwarding functions [2]. Control function of the network device is configured and managed by SDN controllers. SDN switch [3] only provides a simple data forwarding function. Using OpenFlow protocol [4], SDN controller [5] can get information about the state of SDN switches and can dynamically add, delete, update the flow table entries for the switches. SDN controller can control the physical network devices and capture the whole network topology and physical network resource information.

With the rapid development of cloud computing, cloud service providers need to provide a large number of tenants with isolated and quality assurance virtual networks, namely network as a service (NaaS) [6]. Tenant may be a single customer or customer organization who requests the resources from data center network [7]. Virtual Tenant Network (VTN)[8] is an extension technology on OpenDaylight [9] controller that can be used to build a multi-tenant virtual network in cloud data center network scenarios. Virtual network constructed by VTN technology can automatically map to the underlying physical network. In fact, internal communication of virtual network refers to internal communication of the physical network virtual network mapping. Therefore, how to map the SDN virtual network into the physical network is the core technology of VTN. Virtual network mapping problems involve node mapping and link mapping [10]. This paper focuses on node mapping problem.

Currently, VTN provides two node mapping methods including port mapping and VLAN mapping. Port mapping refers to the mapping from virtual interface of VTN virtual switches to port number of the physical switch. The forwarded data from a physical switch port will be regarded as the forwarded data from a virtual interface of virtual network. VLAN mapping refers to the mapping according to the vlan_id value of data frame header. All data frames with the same vlan_id belong to the same virtual network. Although these two mapping methods are relatively mature, they have difficulty in joining a specific physical host to virtual networks. Thus, a host's MAC address-based mapping method is proposed in this paper.

Experimental results show that the host's MAC address-based mapping method simplifies the VTN mapping mechanism and increases the reliability of the VTN.

2. DESIGN OF THE NODE MAPPING ON HOST'S MAC ADDRESS

Fig. (**1**) shows the major components of VTN. VTN includes VTN manager and VTN coordinator. VTN manager is deployed inside SDN controller in the form of plug-in which manages the information of the virtual network including network topology and mapping information. VTN coordinator coordinates multiple SDN controller and provides REST APIs [11] for VTN application. VTN applications are network applications deployed in the virtual network.

A virtual network [12] built by VTN technology consists of virtual nodes(VN), virtual links (VL) and virtual interfaces (VI) on virtual nodes. Virtual nodes (VN) include virtual switch (VS), virtual router (VR) and virtual tunnel (VT), *etc*.

1874-110X/15

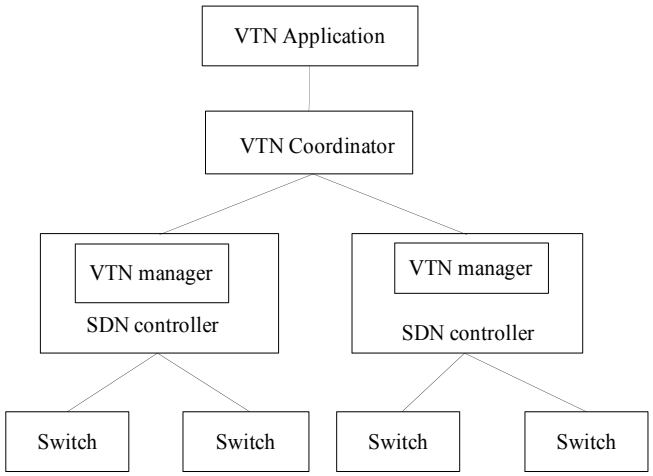

Fig. (1). VTN architecture.

This section introduces the mapping method based on the host's MAC address, which gets the MAC address of the host, then maps the MAC address of the host with the virtual switch, and sends the mapping information to VTN manager. VTN manager then calls the appropriate module interfaces of SDN controller which calculates the path and issues the forward rules to flow table of underlying physical switch to complete VTN internal forwarding.

The sequence diagram of virtual node mapping flow based on host's MAC address is shown in Fig. (**2**).

SDN controller provides host track (HostTrack) module to record the host information in SDN network. Firstly, log on North interface to resolve the host information recorded by HostTrack and extract the MAC address of the host for storage. Secondly, according to tenant requests, the mapping relationships are built between host MAC address and virtual node which include the host's MAC address, switch information connected to a host, and virtual node information. Finally, VTN manager stores these mapping relations, calls the routing module API of SDN controller to calculate the corresponding path and calls flow table module API to add flow table forwarding entries which are issued to the physical switch.

3. IMPLEMENTATION OF THE NODE MAPPING BASED ON HOST MAC ADDRESS

Implementation platform of VTN mapping uses the open source SDN controller OpenDaylight. By mininet tool, physical network topology including switches and hosts is created.

In order to achieve mapping method based on host's MAC address, the host MAC address needs to be obtained firstly. Then, add MacMap mapping method to the mapping lists and design MacMap class to record the MAC address of the hosts. Details are as follows.

(1) Get host information

The North interface module HostTrack of OpenDaylight controller records the host information including IP address, MAC address, vlan-id, etc. In myeclipse development platform, import OpenDaylight controllers including Host-

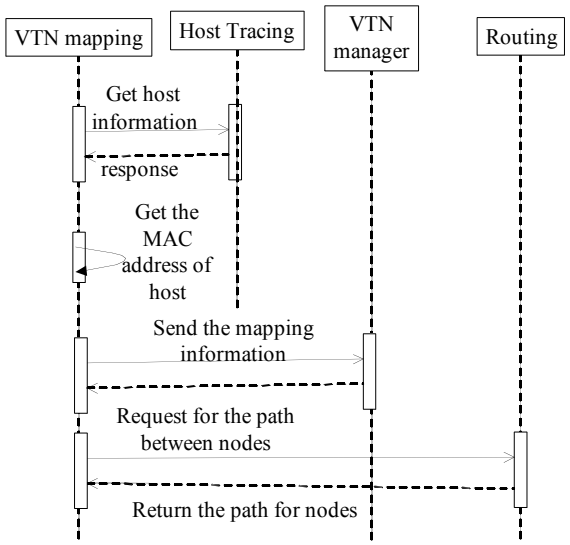

Fig. (2). Sequence diagram of virtual node mapping based on host's MAC address.

Track.jar package and resty kit which provides a simple HTTP/REST client for accessing the north interfaces of OpenDaylight using authentication information. Using the client resolver function, host information in HostTrack is resolved. The directory where the information is saved is hosttracker/default/hosts/acive. The resolved data is saved in the form of JSON object. Code snippets are as follows.

```
URI odl =
URI.create("http://10.10.101.174:8080/co
ntroller/nb/v2/");
       // IP addresses of OpenDaylight
controller
Resty client = new Resty();
          //use Resty to establish client
client.authenticate(odl, "admin",
 "admin".toCharArray());
             //enter authentication data
JSONObject response = cli-
 ent.json(odl.resolve("hosttracker/defau
 lt/hosts/active")).toObject();
//Verification pass, parse the infor-
   mation recorded in north interface
                              module.
```

Extract the MAC address information of each host stored in the form of DataLinkHost class, DataLinkHost class contains the following information.

Host ID	Mac Address	Node ID	Port ID

(2) Establish a virtual network

Point out the MAC address for the mapping which will constitute a virtual network. Call DataLinkHost class which represents the host MAC address information. Use the class as the type to point out the MAC address for the distribution and not for distribution. The definition of MacMap is as follows.

```
Public class MacMap implements Serializ-
 able
{
   Private final
Set<DataLinkHost>allowedHosts=New
 HashSet<DataLinkHost>();
Private final
Set<DataLinkHost>deniedHosts=New Hash-
 Set<DataLinkHost>();
}
```

(3) Virtual Node Mapping

When calling VTN interface to create a virtual node (VN), call the method getAllowedHost() to get a host MAC address for mapping and physical switch (nodeID) used for mapping. In this case the physical switches and virtual switches will be associated. The mapping information including VTN ID, VN, MAC Address, nodeID is stored in the VTN manager. Finally, VTN manager stores all topology information including physical nodes, the virtual node and host MAC address. When the physical switch receives a packet, it checks the MAC address of the packet, sends a packet-in message to the controller, and requests packet forwarding rules. The controller queries VTN manager to know which virtual network the packet belongs to and issues flow table in accordance with the internal virtual network forwarding rules.

4. TEST AND ANALYSIS

4.1. Test Environment

In order to provide virtual network for tenants in SDN network, node mapping mechanism based on host's MAC address is tested and analyzed. Run ubuntu12.04 system built on a physical host server test environment, configure java environment and install OpenDaylight controller helium version. Use simulation tool Mininet for constructing data forwarding network.

OpenDaylight controller runs on the server side. Mininet forwarding network is created including four hosts and three OpenFlow switch. Close default network forwarding application of OpenDaylight controller to make four hosts unreachable to each other.

As shown in Fig. (**3**), node mapping based on MAC addresses can establish the appropriate virtual network for different tenants.

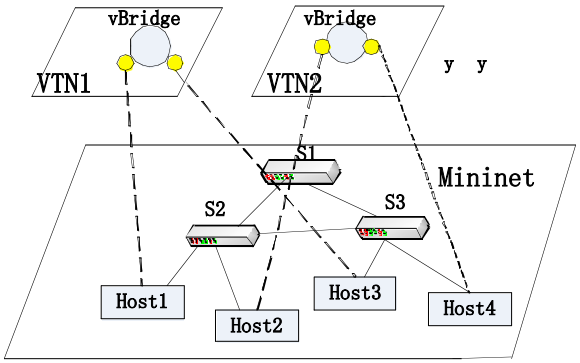

Fig. (3). VTN test network.

4.2. Test Process

Firstly, establish the physical network topology with Mininet. Command of establishing a network is as follows. Close the simple forwarding function inside the controller and each two hosts are unreachable.

```
   Sudo mn --controller=remote,
ip=<controller-ip> --topo tree, 2
```

Secondly, open command terminal of Ubuntu system and call REST APIs for VTN to create two virtual network, named VTN1 and VTN2. VTN1 contains host1 and host3. VTN2 contains host2 and host4. In the establishment of a virtual switch (vBridge), specify the Port mapping for each virtual interface of the virtual switch. The code snip is as follows.

```
   //Create VTN:
   Curl -X POST –d
   '{"vtn":{"vtn_name":"vtn1"}}'
http://172.16.168.51:8080/vtnwebapi/vtns
.json
   //create vBridge:
   curl -X POST -d
   '{"vbridge":{"vbr_name":"vbr1","contro
ller_id":"CONTROLLER1","domain_id":"(DEF
AULT)"}}'
http://172.16.168.51:8080/vtnwebapi/vtns
/vtn1/vbridges.json
   //create virtual switch
   curl -X POST -d
'{"interface":{"if_name":"if1"}}'
   http://172.16.168.51:8080/vtnwebapi/vt
ns/vtn1/vbridges/vbr1/interfaces.json
   curl -X POST -d
'{"interface":{"if_name":"if2"}}'
   http://172.16.168.51:8080/vtnwebapi/vt
ns/vtn1/vbridges/vbr1/interfaces.json
   //create port mapping
   curl -X PUT -d
'{"portmap":{"logical_port_id":"PP-0000-
0000-0000-0004-eth3"}}'
   http://172.16.168.51:8080/vtnwebapi/vt
ns/vtn1/vbridges/vbr1/interfaces/if1/por
tmap.json
   curl -X PUT -d
'{"portmap":{"logical_port_id":"PP-0000-
0000-0000-0006-eth4"}}'
   http://172.16.168.51:8080/vtnwebapi/vt
ns/vtn1/vbridges/vbr1/interfaces/if2/por
tmap.json
```

Use ping command between host1 and host3. The result is shown in Fig. (**4**). The VTN establishment is successful based on the port mapping.

Thirdly, use MAC address-based host mapping to test. The difference from port mapping is that switch virtual interface need not to create and a host directly is mapped to a virtual switch. The specific command is as follows. The

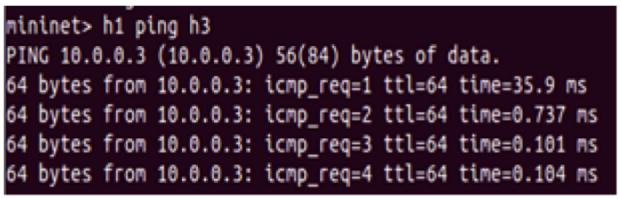

Fig. (4). Host reachability test result.

reachability test result between host1 and host 3 based on MAC address-based host mapping is the same as port mapping.

```
Curl -X PUT
http://localhost:8080/controller/nb/v2
/vtn/default/vtns/Tenant1/vBridge1/macma
p -d'{"host1mac":{....},}'
```

When using the ping test for host1 and host3 within VTN1, two hosts can access each other. The test results show that building a successful VTN and MAC address-based node mapping method is feasible.

Next, in order to demonstrate the reliability of the VTN based on MAC address Host mapping, a Mininet command is shown as follows and used to make the link between s2 and host1 broken to simulate the port of switch s2 is failure. Then use ping command between host1 and host3 based on the port mapping. The result shows that the two hosts are unreachable.

```
mininet> link s1 h1 down
```

Lastly, Mininet not only supports input commands to establish network, but also provides Python API to easily customize the topology by creating a .py file that defines a topology. Add a switch named s4 and connect the host1 with switch s4.

```
class Swicthtopo(topo):
Def_init_(self,n=4,**opts):
Topo._init_(self,**opts)
Switch=self.addSwitch('s4')
self.addLink(host1,s4)
```

According to the test results, when the port fails, port-based node mapping method cannot be used. But by using the MAC address-based node mapping method, through adding the host to the other switches, connectivity among hosts in virtual network will not be affected.

CONCLUSION

Based on the study of SDN-based VTN technology and the node mapping mechanism between virtual networks and physical network, node mapping method based on the MAC address is proposed which simplifies the VTN mapping mechanism and increase VTN reliability. Experimental result shows that the scheme is feasible.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work was financially supported by the innovative research joint funding project of Jiangsu Province (No. BY2013095-108).

REFERENCES

[1] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using OpenFlow: a survey", *Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 493-512, 2014.

[2] L. Chen, and Q. Wu, "Based on construction of data center network SDN techniques", *The internet world*, vol. 1, pp. 40-44, 2013.

[3] ONF, *OpenFlow Switch Specification (version 1.3)*. 2012.

[4] M. Jarschel, and R. Pries, "An openflow-based energy-efficient data center approach", In: *SIGCOMM '12 Proceedings of the ACM SIGCOMM*, Helsinki, 2012, pp. 87-98.

[5] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based comparison and selection of Software Defined Networking (SDN) controllers", In: *Computer Applications and Information Systems (WCCAIS)*, Hammamet, 2014, pp. 1-7.

[6] J. Kempf, Y. Zhang, R. Mishra, and N. Beheshti, "Zeppelin-A third generation data center network virtualization technology based on SDN and MPLS", In: *IEEE 2nd International Conference on Cloud Networking (CloudNet)*, San Francisco, 2013, pp. 1-9.

[7] J. Mudigonda, P. Yalagandula, and J. Mogul, "NetLord: a scalable multi-tenant network architecture for virtualized datacenters", In: *ACM SIGCOMM Computer Communication Review,* Toronto, 2011, pp. 62-73.

[8] OpenDaylight.Project.Virtual_Tenant_Network[Online]. Available: https://wiki.opendaylight.org/view/OpenDaylight_Virtual_Tenant_Network_(VTN):Main/, 2013.

[9] OpenDaylight_Project.WhyOpenDaylight[EB/OL]. Available: http://www.opendaylight.org/why-opendaylight/, 2013.

[10] M. F. Bari, S. R. Chowdhury, R. Ahmed, and B. Raouf, "PolicyCop: an autonomic QoS policy enforcement framework for software defined networks", In: *IEEE SDN for Future Networks and Services (SDN4FNS)*, Trento, 2013, pp. 1-7.

[11] L. Li, and W. Chou, "Design and describe REST API without violating REST: a Petri Net Based Approach", In: *IEEE International Conference on Web Services (ICWS)*, Santa Clara, 2011, pp. 508-515.

[12] R. Cohen, K. Barabash, and L. Schour, "Distributed Overlay Virtual Ethernet (DOVE) integration with Openstack", In: *IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, Ghent, 2013, pp. 1088-1089.

Received: June 10, 2015 Revised: July 29, 2015 Accepted: August 15, 2015