Provision of OpenFlow Virtual Private Networks Service on a Core Network

Steven S. W. Lee, Lok In Liang, Ting-Yun Chan
Department of Communications Engineering
National Chung Cheng University, Chiayi, Taiwan
{example8, example5, example9}@104.com.tw

***Abstract*—With the continuous growth of computer networks, the network architectures are becoming bigger and bigger. Therefore, the management of the entire network becomes more complex. Although Software Defined Network (SDN) can provide central management for a whole network, due to the separation of data and control planes, replacing the traditional devices with SDN devices will cost extra money for a company. In addition, SDN is not popular in daily usage, and is usually used only in core networks such as in communication companies. In this paper, we propose OpenFlow Virtual Private Networks Service to provides a rental service for different types of users. We can let the user use their own controller to manage the rental virtual private network. We implemented this function by using a socket program that was built in our master controller to communicate between two controllers. Finally, we tested the proposed service in a physical SDN-based environment using HP 5900 switches and NOX controller. The results show that our core network can provide an efficient virtual private SDN service.**

Keywords: Software Defined Network, Virtual Network, Network Management System.

I. INTRODUCTION

Traditional Virtual Private Network (VPN) is a communication technology that treats decentralized networks as the same virtual network by using Tunneling Protocol. The weakness of VPN is that the network architecture base on Ethernet which cannot have a global view of a whole network. Therefore, the latest network technology in this generation - Software-Defined Network can let us separate the control plane and data plane. The SDN controller can control all the control planes on each switch. When we use SDN, not only can tackle the problem of large network demand in this generation, but also can reduce the difficulty of the network management. Then, increasing the flexibility of management.

The benefit of SDN is that we can separate the control plane and data plane. Then, we can centrally control the network. Unfortunately, when the user network is located in a different region, it is hard to centralize and control the whole network. Therefore, providing a Virtual Private Networks base on the OpenFlow protocol on a core network is necessary in this case. Not only can deal with the problem of management and hardware maintenance but also can reduce the cost of establishing the network architecture when we are treating of users decentralized network environment as a virtual network and joining to the core network as well as users can use their controller to control the virtual rental network.

The communication between SDN devices and controller is using OpenFlow protocol. On the other words, the hardware has to support OpenFlow protocol for supporting to our core network to provide an OpenFlow VPN. There is a huge cost to build this architecture. Providing an SDN service and renting it to others rather than building a network on our own can let the entrance level of SDN become lower. For example, the company point of view, it can centrally manage all the networks of the branch office. The school point of view, each department or administrative unit can manage their network when we place the OpenFlow switch in each building. Finally, all the renter no need to maintain the hardware of this network.

The rest of this paper is organized as follows. Section II presents a review of the current literature. In Section III, presenting the system architecture that we use in our core network. In Section IV, we propose some method to tackle the problem when we are building this network. In Section V, simulations and analysis results are presented. We conclude of the paper in Section VI.

II. LITERATURE REVIEW

FlowVisor and ADVisor are very in use for building an OpenFlow Virtual Network. FlowVisor is a transparent proxy placing in between OpenFlow switch and controller. FlowVisor can let several users logical topology include all together under a physical topology that has the same links and nodes. Therefore, the cutting of the topology by using FlowVisor is highly restricted under the physical topology which means that the nodes and links have to exactly equal between physical and logical topology when we are using FlowVisor.

ADVisor also places in between OpenFlow switch and controller just like FlowVisor. There is a difference to FlowVisor and ADVisor; ADVisor cannot act as a transparent proxy in between OpenFlow switch and controller. But it can directly completely define a logical topology under the physical topology. For instance, the right-hand side of Figure 1(b), FlowVisor cannot be implemented, but ADVisor can.

Y. C. Chung propose a Design of Bandwidth Guaranteed OpenFlow Virtual Networks Using Robust Optimization [1]. He used ADVisor to implement the OpenFlow Virtual Private Network. The mapping between physical and virtual network are using VLAN tag to distinguish users.

S.Hong and J. P. Jue propose a Survivable Virtual Topology Design in IP over WDN Multi-Domain Networks [2] for minimizing total network link cost. They propose hierarchical Software-Defined networking (H-SDN)-based

architecture to implement it. Also, they propose heuristic approaches that are using partition and contraction mechanisms (PCM) on the virtual topology.

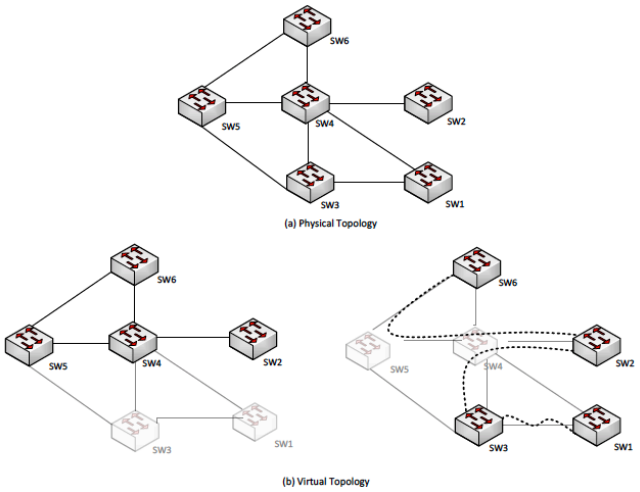


Figure 1: Physical topology and Virtual Topology

In this paper, we would like to use ADVisor to build our OpenFlow Virtual Private Network and use VLAN Tag to distinguish users.

III. System Architecture

The main architecture is illustrated in Figure 2. The middle part of the figure is the core network and the upper and lower parts are different users' virtual networks [3].

The core network is based on SDN [4] architecture. There are three essential systems inside it: Master Controller, Network Management System, and User controllers. These essential systems can be described as follows.

Master Controller (MC): Controlling the entire physical core network and communicating with the user controller

Network Management System (NMS): Recording the mapping information between physical and virtual networks

User Controllers (UC): Controlling the rental virtual private networks

There are two views of the entire architecture.

1. The view of the user (virtual private network): The user view of the rented virtual network includes virtual OpenFlow switches (VS) and UC. Users can use their controller to control their rented virtual network and their own SDN network.
2. The view of the core network: The global view of the whole network includes physical OpenFlow switches (PS), MC, UC and NMS. It contains all the information about users such as user virtual topologies, bandwidth, flow entries and mapping details. Specially, although the user can directly control their rental private network in their view, UC directly connects to MC rather than connecting to all physical switches.

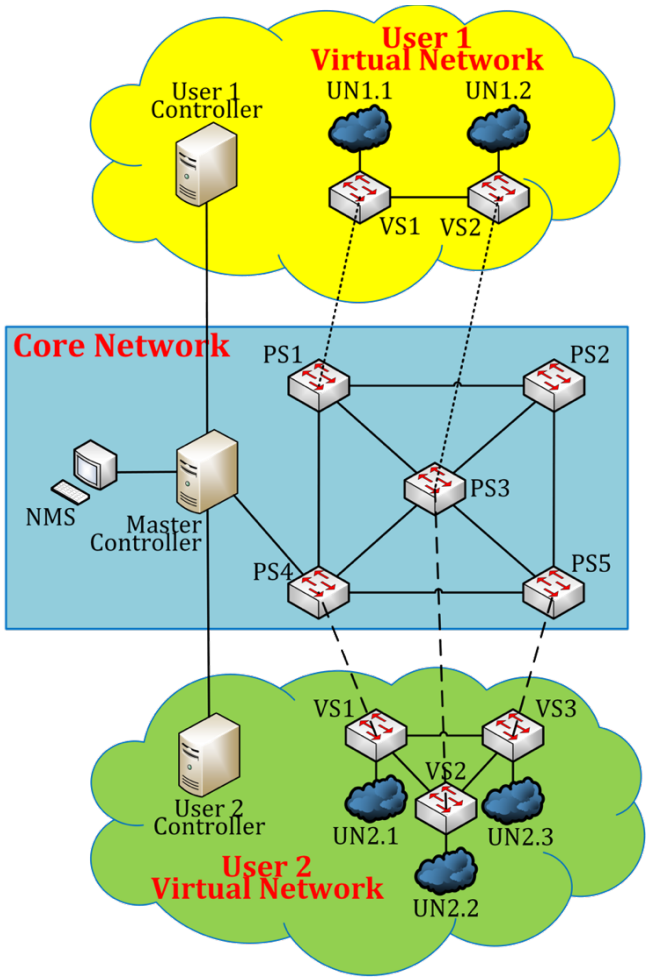


Figure 2: The mapping between physical and virtual networks (NMS = Network Management System, PS = Physical Switch, VS = Virtual Switch, UN = User Network)

As shown in Figure 2, since MC, UCs and NMS communicate through the TCP protocol, these three systems are connected to the ISP network. There are two main parts in this figure. Firstly, the blue block is the core network, including five physical OpenFlow switches which are located at five different locations, one MC which is connected to a physical OpenFlow switch by in-band connection, one NMS and two UCs. Secondly, the yellow and green clouds are two different users' virtual private networks. User 1's virtual network (yellow), contains one UC and two virtual OpenFlow switches, VS1 and VS2, which are mapped to physical switches PS1 and PS3, respectively. Moreover, the virtual switches are connected to the User 1 owned networks (UN1.1 and UN1.2) and controlled by the UC. The virtual network of User 2 (green), consists of one UC and three virtual OpenFlow switches VS1, VS2, and VS3 which are mapped to physical switches PS4, PS3, and PS5, respectively. Note that although those users can only see their own networks in their view, they share the same physical switch (PS3) on the core network.

IV. Implementation

To provide network service on an SDN platform, there are several problems we need to solve. Including communication between UC and MC, the mapping between

physical and virtual, and user rental bandwidth control. Unfortunately, communication between two controllers is not defined in the OpenFlow protocol; therefore, we implement a socket program that simulates the behaviors of OpenFlow switches to tackle the communication between UC and MC. Moreover, we create a data structure for recording the physical-virtual mapping information. Lastly, we use the meter function inside the OpenFlow switch to constrain the bandwidth of each user.

A. Communication between two controllers

The most important part of this system is how a UC controls their rental virtual network through the MC on the core network. Before we introduce the communication between MC and UC, we firstly illustrate the communication between controller and switches.

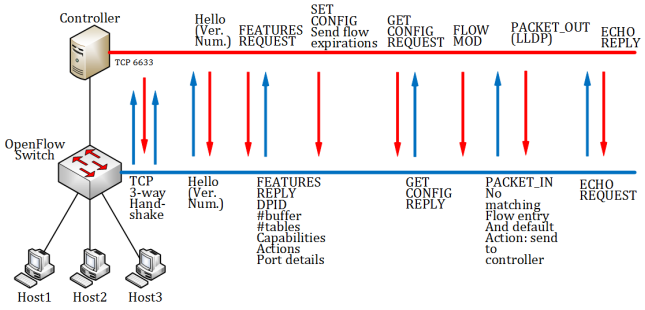


Figure 3: Establishing connection between OpenFlow switch and controller

Communication between controller and switches are defined in OpenFlow protocol [5]. First of all, since they communicate through TCP, they establish a TCP connection by TCP three-way handshake. After that, the switch will actively send a Hello message to the controller to determine the highest supported OpenFlow version. The controller will respond with the OpenFlow version and send a Feature request back to the switch. Next, the switch will respond to a Features reply which includes the switch DPID, buffer size, the number of tables, and supported functions (flow stats, table stats, port stats ... etc.) to the controller. Afterwards, the controller will send a Set Config packet to the switch to configure the behaviors of IP fragments (normal, dropped, or reassembled). Sometimes the controller will ask for a configuration parameter again by sending a Get Config request to switch. The switch must reply a Get Config reply back to the controller at that moment. After finishing all these steps, they can execute the action of Flow Mod, Packet In, Packet Out ... etc. through this OpenFlow connection. Furthermore, the switch sends an Echo Request to the controller to confirm whether it is available or not at that time. If the controller is available, it sends back an Echo Reply to the switch. The steps in the process of establishing the connection and exchanging information between controller and switches are shown in Figure 3.

Unfortunately, OpenFlow protocol does not support communication between two controllers. For this reason, we implemented a socket program inside the Master Controller to simulate the connection behaviors of OpenFlow switches. This socket program actively connects to the user's controller through TCP port 6633. Simulating all the OpenFlow switch request and reply packets and communicating to user controller will be available in this program, as shown in Figure 4. In addition, when a user topology includes three virtual switches, the socket program must open three different socket threads for each virtual switch. Each thread in MC is just like an OpenFlow switch in the user view.

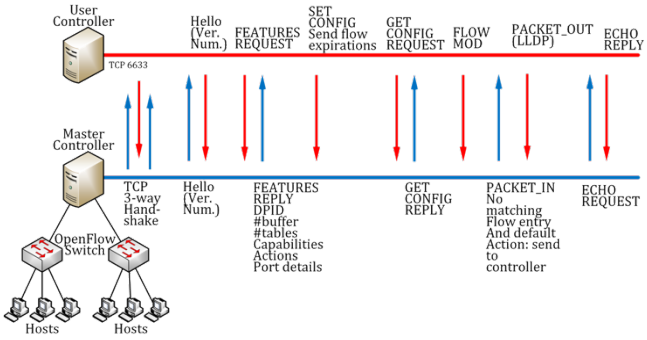


Figure 4: Establishing connection between two controllers

B. Virtual Network Management System

A management system is necessary for our service to record the matching information between the physical and virtual network. When a new user joins our service, it is impossible to pause the MC and reallocate the resources in the core network. Moreover, since different users have different needs of their virtual private network, we cannot pre-allocate the resources for future users. Therefore, the NMS is a crucial management system for registering the demand of different users such as the topology of user virtual private network, the total demand for bandwidth, the number of ports in each virtual switch, etc.

Although we have an NMS for collecting the user information, we must have a structure to record and sort the information. As a result, we create a data structure for saving the physical-virtual matching information. For instance, switch, port, and link mapping. According to Table I, assume that User 1 requires two virtual OpenFlow switches with customized topology, and requires three ports on VS1 and four ports on VS2. On the other hand, User 2 requires three virtual OpenFlow switches with customized topology, requiring five ports on VS1, and four ports on VS2 and VS3.

TABLE I. THE DATA STRUCTURE OF PHYSICAL AND VIRTUAL MAPPING INFORMATION

	Physical-Virtual Mapping[1]				
	Switch Mapping		Total	Port Mapping	
User	VS	PS	Ports	VP	PP
1	1	1	3	1	3
				2	4
				3	2
	2	3	4	1	2
				2	1
				3	4
				4	5
2	1	4	5	…	
	2	3	6	…	
	3	5	4	…	

The switches and ports, both physical and virtual, are recorded and matched by this structure, as shown in Table I. The mapping between physical switches and virtual switches, physical ports and virtual ports is shown for each user. For example, VS2 in User 1 and VS2 in User 2, which are both mapped to PS3, include four and six ports, respectively. Furthermore, the mapping between physical-virtual switches and ports are made according to user requirements for the physical switches and the user-customized topology. In the last two columns of this table, the physical-virtual port mapping table, some parts are highlighted in yellow. A port which is connected to another virtual switch is recorded in the yellow highlight. On the other hand, a port which is connected to user computer or network is recorded in white. This is shown in Figure 5, where the symbol VS1(PS1) means the virtual switch VS1 is mapped to the physical switch PS1. Port number 1(3) means the virtual port 1 in VS1 is mapped to physical port 3 in PS1.

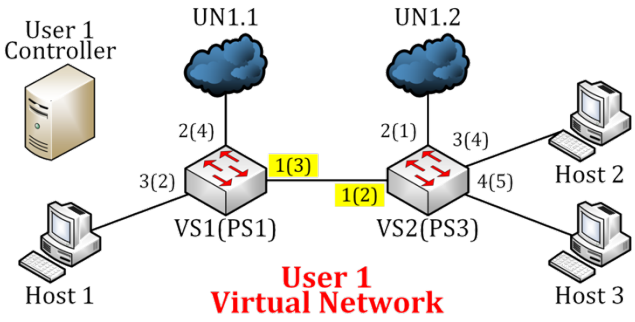
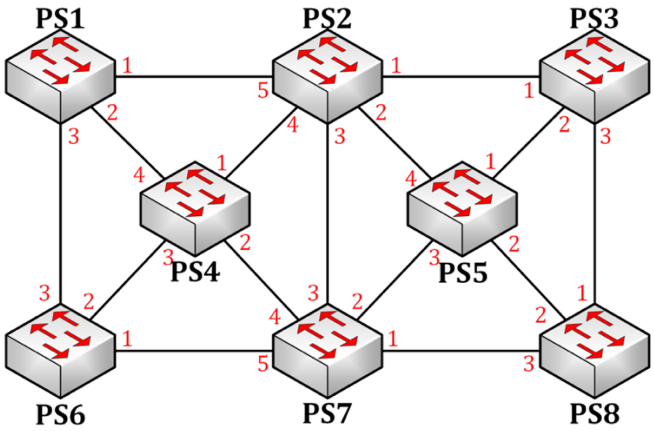
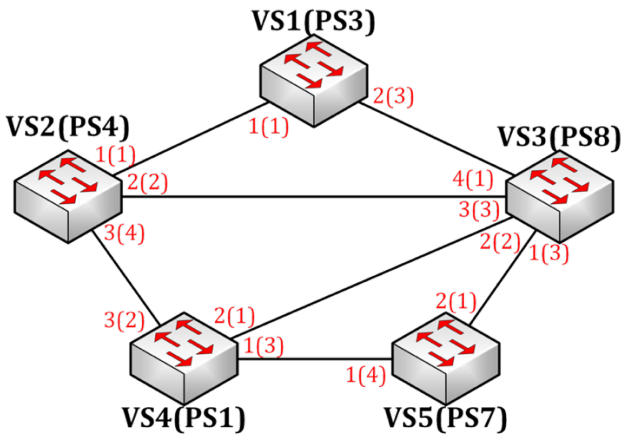


Figure 5: The full view of User 1 Virtual Network with port numbers



(a) Physical topology with port numbers



(b) Virtual topology with port numbers

Figure 6: Physical and virtual port mapping

Since the user topology is only a logical topology, virtual ports and links are not as same as physical topology. Therefore, we demonstrate the ports and links between physical and virtual by using Figure 6.

First of all, we will describe the mapping of ports. According to Figure 6(b), suppose that VS5 would like to connect to VS4, so VS5 must be connected to VS4 via virtual port 1. When we map this virtual port into physical topology, we calculate the shortest path (PS7→PS4→PS1) between PS7 and PS1. Then, according to physical topology information, we can find out the output port from PS7 to PS4 is port 4. Therefore, from VS5 to VS4 must use physical port 4 in PS7. Thus, the physical-virtual port mapping table is generated by this approach.

Secondly, we use Dijkstra's shortest path algorithm to calculate the links between physical and virtual networks. Next, MC will add flow entries which depend on the calculation to specific switches. For example, according to Figure 6(b), suppose that a user would like to create a path which is VS3(PS8)→VS5(PS7)→VS4(PS1) in the user virtual network. We separate this path into two parts, PS8→PS7 and PS7→PS1. Firstly, we use Dijkstra's algorithm [6] to calculate the shortest path (PS8→PS7) from PS8 to PS7 and add the flow entries to these two switches. Finally, we calculate the shortest path (PS7→PS4→PS1) from PS7 to PS1 and add the flow entries to these three switches. If a user would like to create a path which includes four switches, we separate this path into three parts and use the same procedures as in the previous example. In conclusion, this approach can make sure that all virtual paths can correctly map into the core network.

Separating different users on a core network is very important for service in a virtual private network. If the MC cannot divide each user, it will cause the network to be unable to reach a destination or lead to interrupted service on the core network. Therefore, we use VLAN [7] to separate all users. When a user adds a set of flow entries to his virtual switches, we will translate these flow entries into a physical version and add a VLAN tag matching field to it. After that, we will push (go in from a port) or pop (go out from a port) a VLAN tag header in front of a packet on the white highlight ports and match VLAN tag ID with switch flows table on the yellow highlight ports. For instance, as shown in Figure 5, assume that User 1 would like to send something from Host 1 to Host 2, User 1 will add a flow entry in VS1 and VS2. Then, MC will add VLAN tag ID matching field which is equal to 1 (User 1) on it. Next, when a packet is send out from Host 1, this packet will be pushed a VLAN tag header on port 3 of VS1. Then, port 1 of VS1 and VS2 just match the VLAN header with the switches flow tables. Finally, this packet will be popped the VLAN tag header on port 3 of VS2. Therefore, VLAN can simply help us to separate all users correctly.

C. User bandwidth control

All users share the same resource of bandwidth on a core network. So, constraining the bandwidth for each user is necessary. In OpenFlow protocol, it can monitor a user bandwidth by adding a constraint on a flow entry. When the user current bandwidth exceeds the required bandwidth, the excess bandwidth will be dropped by this constrained flow

entry. As a result, not only is all users' required bandwidth guaranteed but users will be prevented from misuse of network resources.

V. Simulations and Analysis

We analyze our service in a physical environment. We illustrate the experiment architecture in Figure 6(b). We use two HP5900 OpenFlow switches which separate eight OpenFlow VLANs to construct our environment. Additionally, each OpenFlow VLAN is just like a switch in this architecture. We connect at least one computer to each switch to simulate the condition of the actual environment. Finally, we implement the MC (by using NOX [8] controller) and NMS on the same computer. Then, we connect this computer to the ISP network, therefore, UC can connect to the MC through the ISP network.

A. User Virtual Network

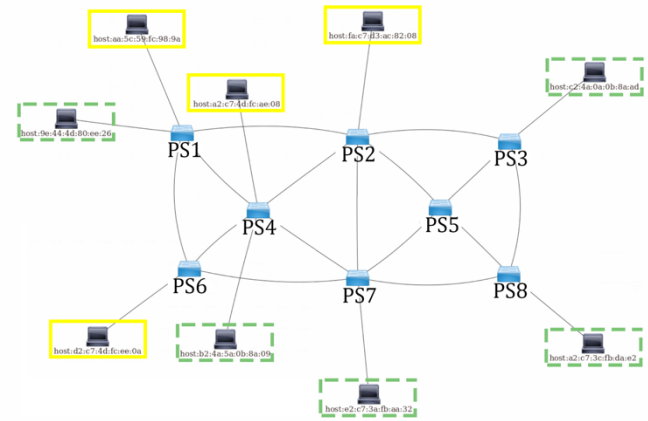


Figure 7: OpenDaylight GUI topology of the core network

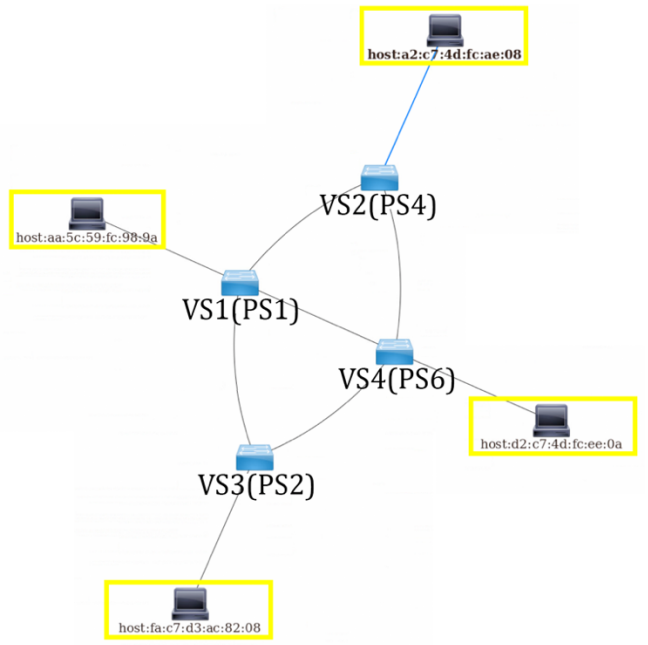


Figure 8: OpenDaylight GUI topology of User 1 virtual network

As shown in Figure 7-Figure 9, the topologies on the OpenDaylight [9] GUI console prove that our service works. Since OpenDaylight controller provides a real-time topology function, we use this function to test our service. First of all, the core network in Figure 7 includes two different groups of computers with different frame colors, yellow solid and green dotted, which are User 1 and User 2 computer groups, respectively. User 1 and User 2 topologies are shown in Figure 8-Figure 9. Considering the complexity of these three figures, it is remarkable that MC and UC can successfully establish a connection to exchange information through our socket program and also virtual network management system can successfully map the nodes, ports, and links between physical and virtual. Additionally, the core network has a global view of the whole network while users only have their private network view.

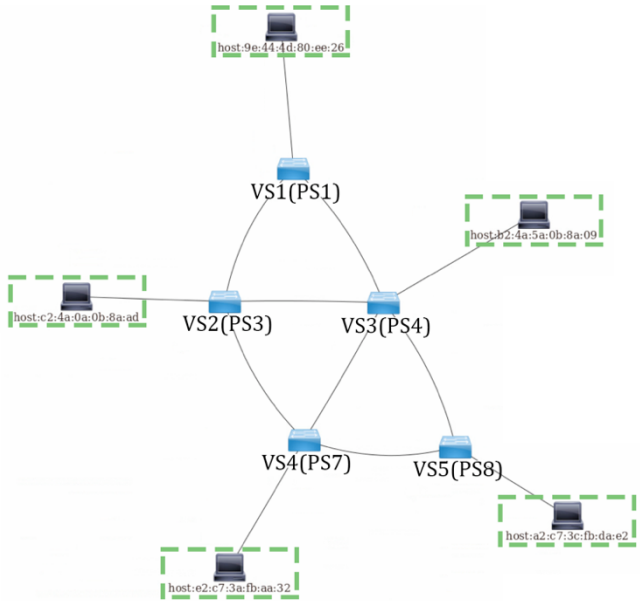


Figure 9: OpenDaylight GUI topology of User 2 virtual network

B. User Bandwidth control

We propose to use OpenFlow meter for bandwidth control of each user. In the following experiment, we use two computer groups from User 1 and User 2. We send traffic from source to destination in these computer groups at the same time by using the iPerf software. The experiment bandwidth data is shown in Table II, where User 1 and User 2 requested bandwidths are 300 Mbps and 200 Mbps, respectively.

TABLE II. The Experiment Data for Testing Bandwidth Control

User	Bandwidth	
	Requested	Testing
1	300 Mbps (38MiB/s)	700 Mbps (88 MiB/s)
2	200 Mbps (25MiB/s)	500 Mbps (63 MiB/s)

The concept of this experiment is shown in Figure 10. Firstly, User 1 host 1 (U1H1) sends 700 Mbps (88 MiB/s) traffic to host 2 (U1H2). Since User 1 only required 300 Mbps (38 MiB/s) bandwidth in his virtual network, all the flow entries which are added by User 1 are subject to a bandwidth constraint. Therefore, when the traffic passes

through PS2, it will drop the traffic back to the requested bandwidth, as shown in Figure 11. As with User 1, User 2 host 1 (U2H1) sends a 500 Mbps (63 MiB/s) traffic to host 2 (U2H2). Then, PS4 will drop the traffic back to 200 Mbps (25 MiB/s), as shown in Figure 12.

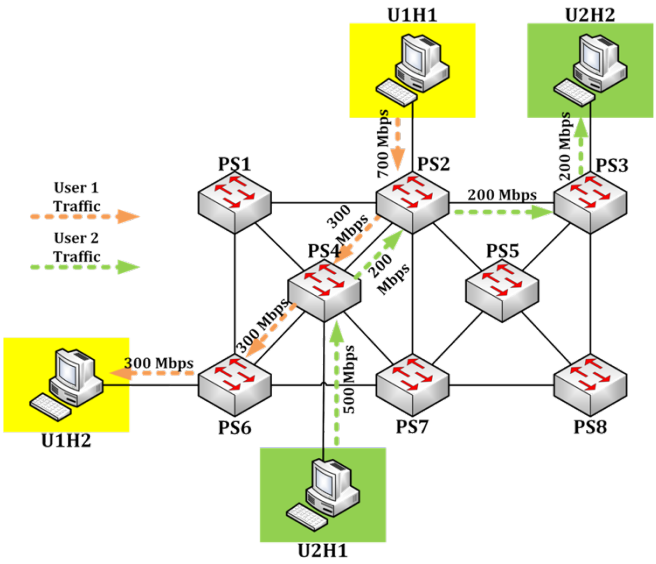
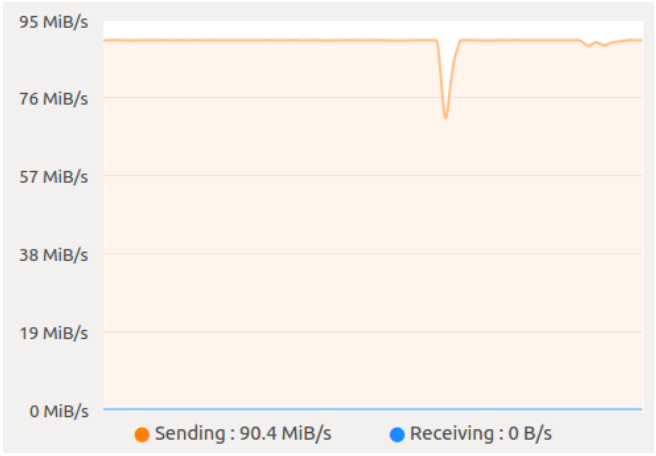
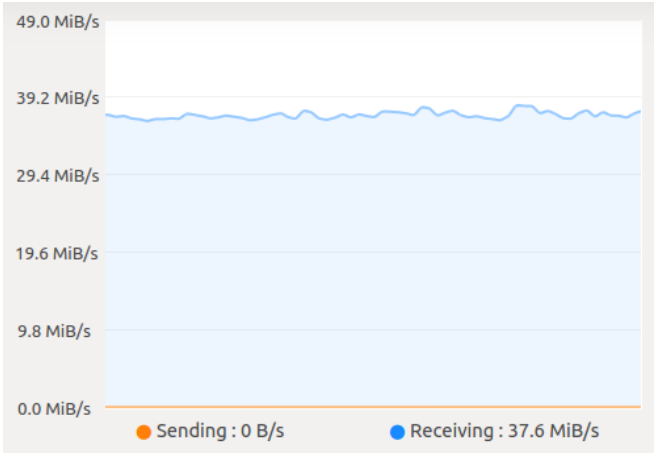


Figure 10: The experimental environment of bandwidth testing

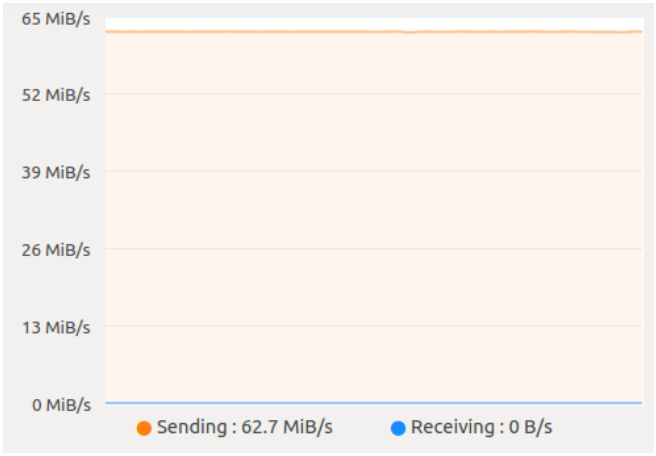


(a) Traffic sent from host 1

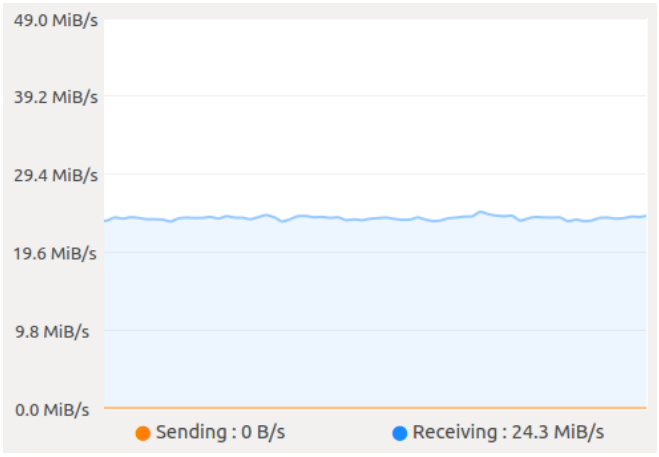


(b) Traffic received by host 2

Figure 11: The traffic situation for User 1



(a) Traffic sent from host 1



(b) Traffic received by host 2

Figure 12: The traffic situation for User 2

Finally, we add an OpenVswitch [10] between PS2 and PS4 to observe the bandwidth on the core network. As shown in Figure 13, the sending traffic (24.4 Mib/s) is the traffic of User 2 and the receiving traffic (37.8 MiB/s) is User 1. In conclusion, not only we can limit the bandwidth to each user by using meter function, but also we are able to separate different users by using VLAN Tag.

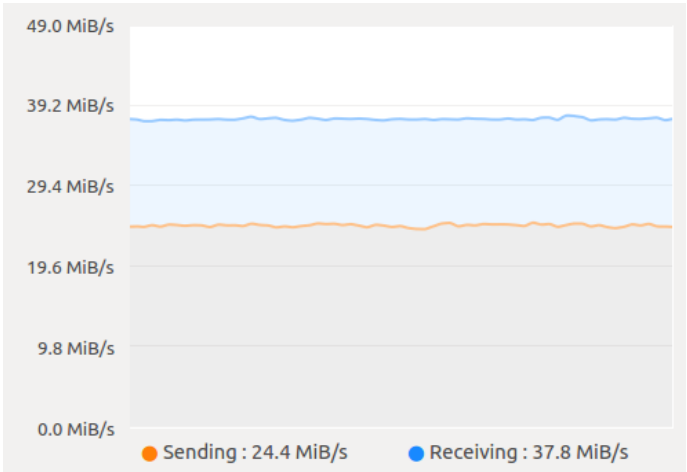


Figure 13: The traffic situation on the core network

VI. Conclusion

This paper presents an approach to communication between two controllers, virtual network management system and bandwidth control by implementing an OpenFlow virtual private network on a core network. We implemented a socket program in the MC to simulate the communication of an OpenFlow switch. For each user requirement, we created an NMS with an extraordinary data structure and approaches to manage each user in the core network. Finally, we used VLAN Tag and OpenFlow meter to separate different users and limit the bandwidth of each user, respectively. In this paper, we evaluated our service using an HP5900 OpenFlow switch. Experimental results showed that our service can provide a virtual OpenFlow private network for different users. In future work, we will consider proposing an optimization approach for allocating user resources.

References

[1] Y.-C. Chung, "Design of Bandwidth Guaranteed OpenFlow Virtual Networks Using Robust Optimization," 2014.

[2] S. Hong, J. P. Jue, P. Park, H. Yoon, H. Ryu, and S. Hong, "Survivable virtual topology design in IP over WDM multi-domain networks," in *2015 IEEE International Conference on Communications (ICC)*, 2015, pp. 5150-5155.

[3] Z. Aqun, Y. Yuan, J. Yi, and G. Guanqun, "Research on tunneling techniques in virtual private networks," in *Communication Technology Proceedings, 2000. WCC-ICCT 2000. International Conference on*, 2000, vol. 1, pp. 691-697: IEEE.

[4] N. McKeown *et al.*, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review,* vol. 38, no. 2, pp. 69-74, 2008.

[5] B. Pfaff, B. Lantz, and B. Heller, "Openflow switch specification, version 1.3. 0," *Open Networking Foundation,* 2012.

[6] S. Skiena, "Dijkstra's algorithm," *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, Reading, MA: Addison-Wesley,* pp. 225-227, 1990.

[7] P. J. Frantz and G. O. Thompson, "VLAN frame format," ed: Google Patents, 1999.

[8] N. Gude *et al.*, "NOX: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review,* vol. 38, no. 3, pp. 105-110, 2008.

[9] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on a*, 2014, pp. 1-6: IEEE.

[10] B. Pfaff *et al.*, "The Design and Implementation of Open vSwitch," in *NSDI*, 2015, pp. 117-130.