Performance Optimization of Service Chain for vCPE in Small Networks with OpenFlow Switch

Nen-Fu Huang*, Chi-Hsuan Li*, Chia-Chi Chen*, and I-Hsien Hsu*
* Department of Computer Science National Tsing Hua University, Hsinchu, Taiwan
Email: nfhuang@cs.nthu.edu.tw

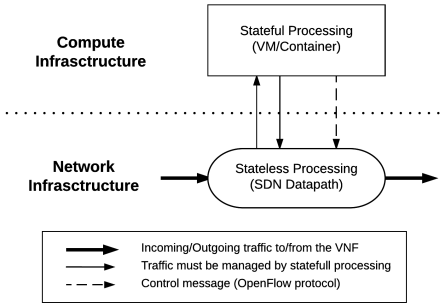

Fig. 1. Overview of network functions.

***Abstract*—**

***Index Terms*—Software defined network, network function virtualization, NFV orchestration, virtual customer premise equipment, OpenFlow, multiple flow tables**

I. Introduction

II. Related Work

III. System Description

A. Overview of Network Functions

Our network functions (Fig. 1) are designed with SDN-enabled NFV architecture concept [1], using the synergies between computer infrastructures (NFVs) and network infrastructures (NFVIs) [2], [3]. An NFV is a VNF controller, mainly used for addressing stateful processing and NFVI is an SDN switch used for stateless processing.

1) Stateful Processing Component (VNF Controller in VM or Container): This component is used to control the workflow, maintain the state associated with the VNF, and provide an interface for service providers or customers to configure and update the behavior of the stateless datapath processing component. We used an SDN controller to implement the VNF controller; and notably, we use southbound APIs of the SDN controller framework to manage the interface between the stateful and stateless components with the OpenFlow protocol, which was originally designed for this purpose.

2) Stateless Processing Component (SDN Datapath): Stateless processing component is implemented by SDN datapath resources and is optimized for data plane traffic processing. Because an SDN switch can be decoupled with control plane and data plane, the switch can accept the control messages from the stateful processing component.

By using the advantages of this architecture, we can assign stateless or light-weight state work to the SDN switch (e.g., packet filtering and packet counting) to reduce the load on the computing resources. If we want to update our service, we are required to update only the stateful component, because the stateless component merely follows the commands from the stateful component.

B. Service Deployment Model

Unlike a related study that explored the virtualization of network function in PE devices [4], we introduced a network function service deployment model based on the NetFATE (Network Function at the Edge) approach [5]; the architecture of the model is presented in Fig. 1. Because computing infrastructures involves algorithms and policies and the generic network devices perform only stateless processing, the customers need to simply purchase a general SDN switch for their home gateway. They can obtain a different network function service by subscribing to a different VNF controller through our vCPE platform.

Fig. 2 illustrates the service deployment model. Each green area is a local network domain of the customer. An SDN switch is presented at the gateway of this domain. The customer can subscribe to our vCPE service through our dashboard. After subscription, the vCPE system creates a new Docker container or new VM in which an SDN controller is run. The customer only needs to set up the gateway SDN switch to connect the SDN controller through the OpenFlow protocol; thereafter, the switch executes the service.

C. vCPE System Architecture

The architecture (Fig. 3) is adapted from the NFV-MANO architectural framework in [6], including an infrastructure controller, an infrastructure orchestrator, a cloud database, VNF controllers and a VNF Orchestrator. Each component is introduced in the following subsection.

1) Infrastructure Controller: The infrastructure controller comprises a Docker management server that can manage the Docker resources like containers and images and a OpenStack server that can manage the VM resources. The infrastructure controller does not manage customer authentication or maintaining the state of the running service; however, it follows the request from the infrastructure orchestrator to create, delete, start, stop, and inspect containers and VMs.

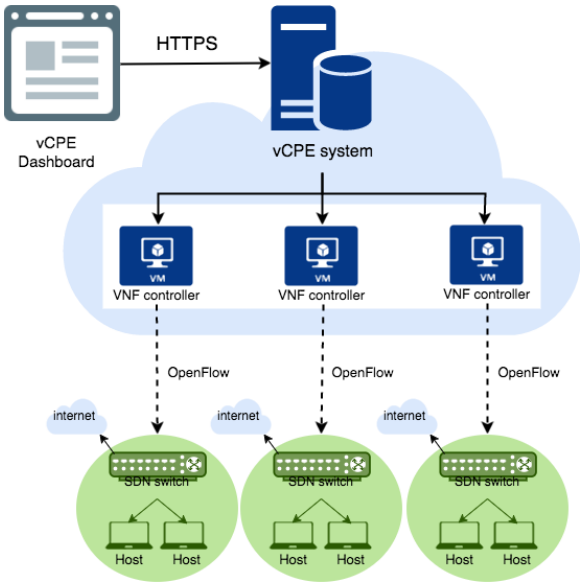

Fig. 2. Service deployment model.

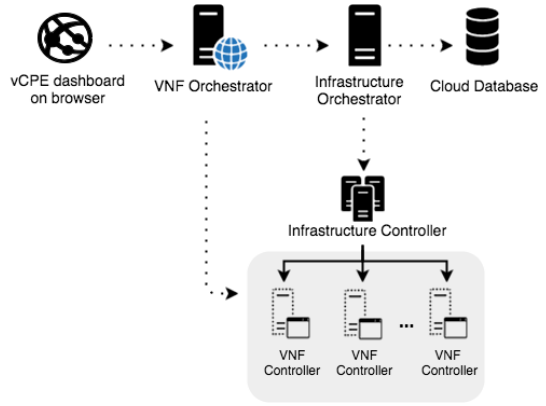

Fig. 3. Architecture of the vCPE framework.

2) Infrastructure Orchestrator: The infrastructure orchestrator plays a key role in our system. It connects and automates the workflows when our services are deployed. When a customer subscribes to our service, the infrastructure orchestrator first authenticates the customer, calls the infrastructure controller to create a container or a VM for the customer, and then updates the information in the database. The infrastructure orchestrator controls the entire life-cycle of our vCPE services.

3) Cloud Database: The cloud database is used for restoring the meta data of our vCPE services, which include each customers credentials, customers container/VM settings, and virtual CPE service states. The cloud database employs PostgreSQL, which is an open source, easily customizable and object-relational database system. Only the infrastructure orchestrator has permissions to access the cloud database.

4) VNF Controllers: VNF controllers comprises an SDN controller developed using Ryu framework [7] and a remote launcher module. The SDN controller does not have a remote launcher module for remotely executing an SDN controller. We built a light-weight server as a launcher module to resolve this problem. The remote launcher module monitors the SDN controller process ID (PID) and kills the SDN controller PID on demand. Once the infrastructure controller creates the container or the VM, the remote module will initially runs, waiting for requests from VNF Orchestrator. The details of an SDN controller design are presented at Section IV.

5) VNF Orchestrator: The VNF orchestrator is a web application server hosted on Amazon web server, and provides to customers an online dashboard for vCPE services management and configuration. Through the web-based UI provided by the VNF orchestrator, customers can subscribe to the desired service without typing any command through the command line interface. After receiving the subscription message, the VNF orchestrator requests the infrastructure orchestrator to create a new VNF controller, and then sends the virtual CPE configuration to the new VNF controller. Based on configuration demands under different conditions, the network administrator can select any of the listed network functions on the dashboard, such as Firewall, NAT, DHCP, and quality of service (QoS) management.

IV. Network Function With Multiple Flow Table Management Model

A. Multiple Flow Tables Strategy

In subsection III-A, we introduce the vCPE service design architecture. The network functions are managed through cooperation between the SDN controller on the cloud and SDN switch at the local network gateway. The controller transforms the network functions into a series of OpenFlow rule requests and sends it to the SDN switch. Following the orders from the controller, the SDN switch inserts the rules into its flow tables, examines the incoming packets against the flow entry match fields, and executes the actions in matching rules. The flow table [8] defines all matching and corresponding processing, thus playing an important role in the executive network function.

We found that a single flow table restricts the implementation of our network functions. In [9], two conditions under which a single flow table is too restrictive were reported. The first is a condition where a single packet must perform independent actions based on matching with different fields. The second is a condition where the packet requires two-stage processing. To satisfy both conditions, we implemented the network functions by using a multiple flow table strategy.

Before we discuss about the multiple flow table strategy, we introduce the pipeline of OpenFlow flow table first [10]. The processing of each packet always starts at the first flow table. When processed by a flow table, the packet is matched the flow entries of the flow table and adds corresponding action to the instruction set. The packet can executes the instrcution set immediately, or execute after finishing the journey in switch. A flow entry can direct a packet to next table by go-to action. In our multiple flow table management model, we will set the go-to-next-table action as the table-miss action. Therefore, the packet is processed table-by-table in a certain sequence.

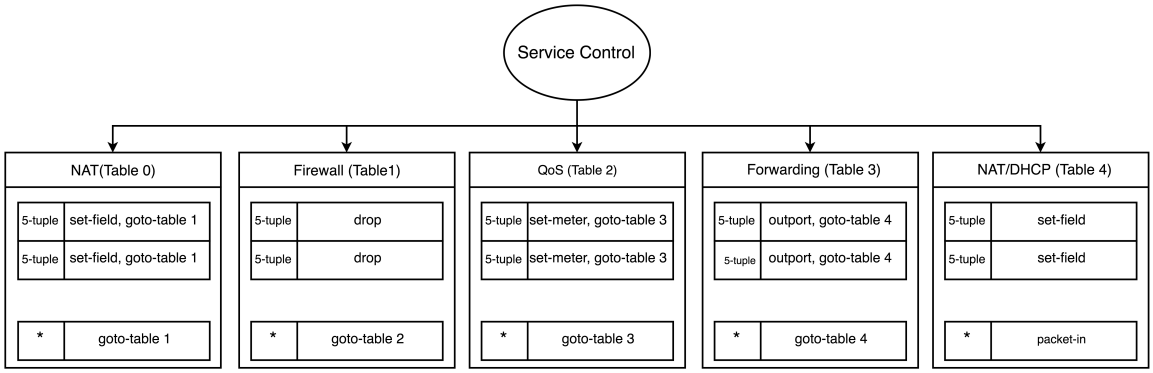

Fig. 4. Flow table order of vCPE service.

In a multiple flow table strategy, it is most important to determine which flow table the rules should be inserted into. We used the network function as a demarcation, that is, SDN applications responsible for specific network functions inserted rules into one specific flow table to enable us to focus on the design of the network function itself. However, the order of the flow table and the sequence of the network functions become crucial. This can be addressed by considering the type of match and action in the rules generated by the network function.

The network functions of vCPE services are the firewall service, NAT, DHCP, forwarding, and QoS. The order of each function was determined as shown in Fig. 4 (note that the flow tables are counted from zero). Each packet In the following sections, we introduce the method of implementing these network functions, the type of rules to be inserted into the SDN switch, and the effect of these rules on deciding the order of the flow tables

B. Service Control

Service control is used to enable or disable services. To enable a service, the table-miss rule should be modified. A packet-in rule is always placed in the flow table of the last active service as a table miss in case there is no corresponding rule. To enable the service chain, the rules of each service except the last service contain an additional action, “go to next flow table”, which enables the packets to continue to pass through all active services.

To disable a service, we must not only modify the table-miss rule but also add an enforce rule. Each enforce rule has maximum priority with the action, “go to next flow table”. It indicates that packets still pass through the disabled services table, but ignore other rules and proceed to the next flow table.

C. Firewall

The firewall service can dynamically block traffic and prevent the packets from causing a packet-in event.

On the dashboard, we can specify the blocking policies. There are 3 kinds of policies:

- block any traffic from a source IP or destination IP address;
- block traffic based on known layer 4 protocols, such as SSH and HTTP;
- block traffic to customize layer 4 ports of a host.

For different policies, the controller applies corresponding rules to the SDN switch. After the policies are set, the blocking rules are immediately installed. Subsequently, any traffic that satisfies the blocking criteria is dropped. Normal traffic is unaffected.

As shown in Table I, all the actions of flow entries are dropped. The first rule illustrates that SSH connection with the source IP address 192.168.2.1 is blocked. The second rule indicates that the flow entry blocks the Telnet protocol.

In our multiple flow table model, the firewall service is located in flow table 1 because once packets are detected by the blocking rules, they do not need to be applied to any other services. The packets that satisfy the blocking rules are immediately dropped, and their journey in the flow table ends. The other unblocked packets pass all blocking rules and finally satisfy the table-miss rule, which allows the packets to proceed to the next flow table. The action of the firewall is different from those of other services, because in other services, irrespective of the actions taken with the packets, the packets must proceed to the next flow table.

D. NAT

The NAT service allows numerous hosts to use one public IP address for connecting to the network. To achieve this, the SDN controller must set the packet header field. Because the SDN switch sets the field, the first packet must proceed to the controller. When the controller adds the flow to the SDN switch, the packet does not proceed to the controller. This can reduce the burden on the controller.

Following is an example that illustrates the modification of the IP address and port number by using the NAT service. For an outgoing packet, the SDN switch does not have any flow entry in the flow table, and hence, a packet-in event is triggered initially. The packet that is sent by a private network host is sent to the SDN controller, and the packet header fields are modified using the set-field action. The source IP address and source port number of the outgoing packet are modified to a public IP address and a new port number is remapped

TABLE I
FIREWALL RULES IN FLOW ENTRY

IP proto	IP src	IP dst	L4 sport	L4 dport	action
TCP	192.168.2.1	*	*	22	drop
TCP	*	*	*	23	drop

Symbol * represents wildcard (matches any value).

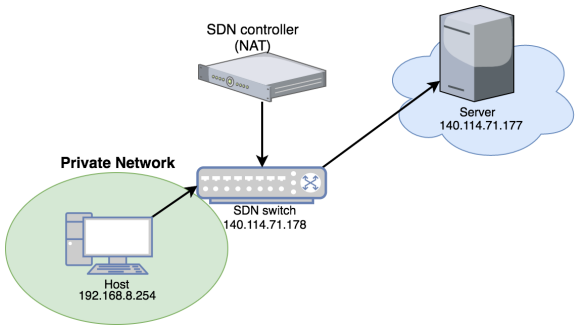

Fig. 5. Example of modification of IP address and port number by NAT service.

for NAT. For the incoming packet, the destination IP address and destination port number are modified to fit the private IP address and port number. Subsequently, the SDN controller adds these flow entries to the SDN switch; all packets must be sent to the controller

In Fig. 5, the public IP address of NAT is 140.114.71.178, and the host private IP address is 192.168.8.254 with the port number 7878. The client sent the packet to a server with the IP address 140.114.71.177 and port number 9898.

As shown in Table II, when the host sends the packet to the server (outgoing), a packet-in event is triggered, and the packet is sent to the controller. The set-field action modifies the source IP address to a public IP address of NAT, 140.114.71.178, and the source port to 2000. When the server sends the packet back to the client (incoming), the packet header field is modified. The destination IP address and destination port number are modified to 192.168.8.254 and 7788, respectively.

In the single flow table framework, two rules must be added to the SDN switch to match the outgoing and incoming situations. First, we predicted that the NAT service must be placed in the last table of the multiple flow table framework because the NAT service must set the packet header fields and enable connection to the outside network. Most importantly, the SDN switch is placed according to the order of tables to match the field. The outgoing and incoming situations must be considered. In consideration of all the aforementioned factors, we place the NAT service in the first and last tables in our multiple flow table framework.

E. DHCP

The DHCP service implements the DHCP protocol to dynamically assign IP addresses to hosts. A DHCP operation uses the UDP protocol. Clients use port 68 as the source port and port 67 as the destination port. By contrast, the server uses port 67 as the source port and port 68 as the destination port. Our system can handle packets to realize the DHCP service.

This service is executed through the following steps:

1) The controller adds a DHCP rule for DHCP packets when the service is enabled.
2) All packets match this DHCP rule, causing a packet-in event.
3) The controller determines whether a packet is a DHCP discovery packet. If so, the controller assigns an IP address, generates a DHCP offer, and then performs a packet-out event. If not, the controller determines whether it is a DHCP request. If the result is positive, the controller generates a DHCP acknowledgement and then performs a packet-out event.

Our system supports multiple flow tables; however, a specific flow table for the DHCP service is not required because only one rule is installed for all hosts who request the DHCP service. When the service is disabled, the DHCP rule is deleted, and the packets continue to pass through our service chain. The subsequent DHCP packets can reach other DHCP servers by forwarding service.

F. Traffic Mirroring

G. Forwarding

In the forwarding service, when the first packet in a new connection is incoming, a packet-in event occurs because no corresponding rule is present. When the controller receives the packet, it records the IP-layer information, including the source IP address, destination IP address, input port number, source MAC address, and destination MAC address. By using the recorded information, the controller can install a 5-tuple forwarding rule with out-port action for this connection, and the subsequent packets do not need to undergo the packet-in event. The 5-tuple comprises the source IP address, destination IP address, network layer protocol, source layer 4 port, and destination layer 4 port.

To gather per-session statistical information, 5-tuple rules are required. Therefore, rules based on the MAC address are not added. The controller installs a pair of dummy rules for every connection, and then requests the switch to obtain current flow statistics every second. Thus, the real-time bandwidth statistics of each connection can be obtained by merely subtracting the byte count from the byte count of the last second.

H. QoS

Quality of Service (QoS) is always used to prevent the network congestion and make sure all of applications are

TABLE II
FLOW ENTRY FOR MODIFYING THE PACKET HEADER FIELDS

	IP src	IP dst	L4 src port	L4 dst port	action
1. outgoing	192.168.8.254	140.114.71.177	7878	9898	IP src → 140.114.71.178; L4 src port → 2000
2. ingoing	140.114.71.177	140.114.71.178	9898	2000	IP dst → 192.168.8.254; L4 dst port → 7878

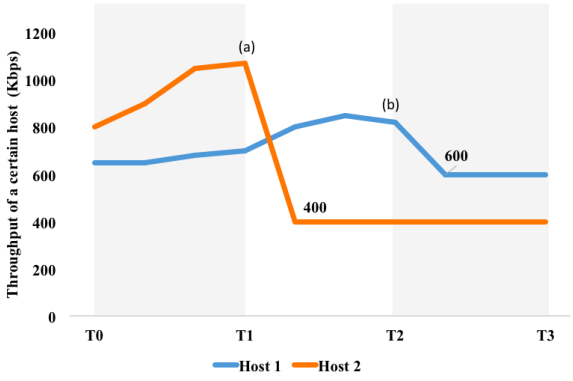

Fig. 6. Illustration of rate limiting for a certain host

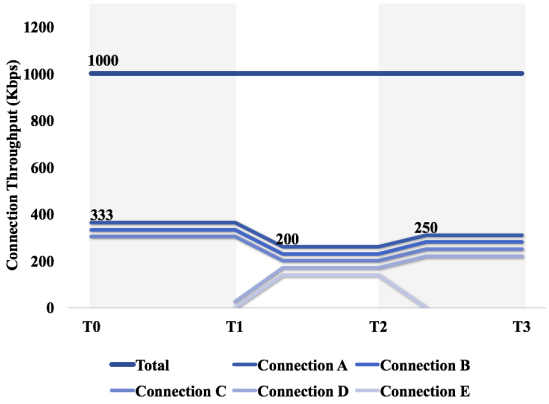

Fig. 7. Illustration of rate limiting for a certain application.

functioning properly. Therefore, we provide two management functions with QoS for the transport of traffic with special requirements. Then we will introduce two strategies of QoS and discuss the order of QoS in the multiple table model.

1) Rate Limitation of Hosts: When some hosts utilize a high bandwidth of network, it will occur the traffic congestion probably. As a result, we are providing the rate limitation for hosts to pervent this. To implement the rate limiting for hosts, we use meter, which is defined within OpenFlow protocol 1.3, to set the limitation in the specific bandwidth. Therefore, we will create a meter for a desired bandwidth and assign the flows of target host with it.

This method is illustrated in Fig. 6. When host 2 utilizes a substantial amount of bandwidth from the network in T1, the network administrator decides to set the rate limitation of host 2 to 400 Kbps. When the controller receives this request, marked as (a), it creates a meter with meter id = 1 and bandwidth = 400 Kbps. Then controller will add the rule with the destination MAC address = MAC address of host 2 and meter = 1.

After the limitation, the rate of host 2 will reduce and be limited to 400 Kbps immediately. A similar situation occurs to host 1 in T2. The administrator decide to set the rate of host 1 in 600 Kbps, marked as (b). Then the traffic from host 1 is limited to 600 Kbps as expected.

2) Rate Limitation of Applications: There are more and more network applications such as online games, video streaming, and conference calls are used. However, it should make sure the important applications are functioning properly. Consequently, we integrated a flow classification engine to identify the flow belongs to which application.

To avoid some applications take up a lot of bandwidth, we can limit the bandwidth for a certain application. We will mirror the packet to flow classification engin and identify which application belongs to. If we want to limit specific application, we just add a 5-tuple rule and meter to those packets which belong to this application. Then the controller will set those flows into the same meter and they will share the bandwidth by this meter.

This method is illustrated in Fig. 7. When the network administrator decides to limit a specific application to a certain bandwidth, the controller will limit all packets of this application with 5-tuple rule. For example, we want to limited specific application to 1000 Kbps and use flow classification enginedetermine to determine which flows belong to this application.

In T0, there are three connections belong to this application, and we will set those connections to the same meter with 5-tuple rule. In T1, there are more two connections are added to this application, and we will also set those new connections to the same meter with 5-tuple rule. In T2, there is a connection stop from this application. Therefore, the bandwidth will be shared by other rest of the connection. However, the sum of bandwidth from this application is always 1000 Kbps.

3) The Flow Table Order of Forwarding and QoS Service: Because the location of NAT, DHCP, and the firewall have been determined, we only need to decide the arrangement of QoS and forwarding. Assume that we place the QoS flow table after the forwarding flow table. In addition, we have only two services enabled, forwarding and QoS; therefore, the packet-in rule is in the last flow table of active service, Qos service. Then, suppose that a host is not limited by QoS policies. The first packet is not affected in both arrangements.

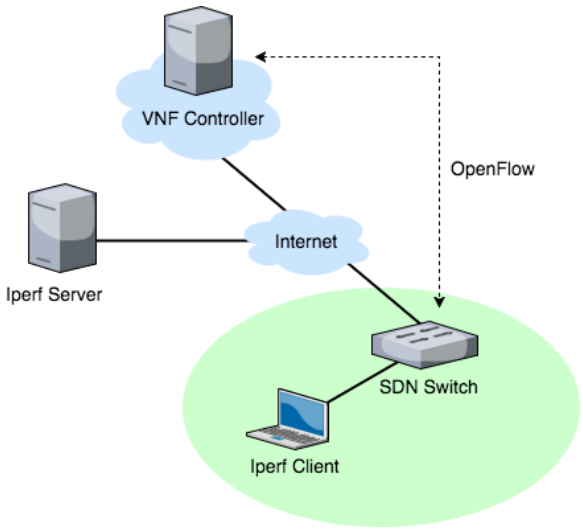

Fig. 8. Multiple Table Performance Evaluation Scenario

Fig. 9. Results of Multiple Table Performance Evaluation

For the subsequent packets, a difference can be observed. The packets that satisfy the rules in the forwarding flow table can not match rate limit rules in QoS flow table, because the host is not limited by QoS service; instead. As a result, the packets cause packet-in events by matching the packet-in rule in QoS service. This is unexpected because the packets already get the out-port action from the forwarding service. That is, it is not necessary to send these packet go to controller, and any packet-in event increases the controllers load.

To reduce this load on the controller, we place the QoS flow table ahead of the forwarding flow table. In this scenario, all packets that pass through the QoS flow table continue to proceed to the forwarding flow table without satisfying any QoS rules. Then, all packets except the first packet are merely forwarded by the forwarding service instead of causing packet-in events. Thus, the controllers load decreases.

V. Performance Evaluation

A. Multiple Table Performance

Implementing a single flow table framework is easier than implementing a multiple flow table framework; however, multiple flow table frameworks are more flexible. To verify the efficiency of the multiple flow table vCPE framework, we conducted an experiment by using the NAT service to compare the throughput between single flow table vCPE and multiple flow table vCPE.

An overview of the experimental environment is presented in Fig. 8. The NFV controller was run on the Dell PowerEdge R630 rack server, and the Pica8 P-3290 switch [11] was used as the SDN switch. We used iPerf2 [12] to generate network traffic. The iPerf server connected the public IP address with 140.114.71.178, and then the iPerf client connected to the SDN switch and was controlled by the NFV controller. The NFV controller ran the NAT service, but used different frameworks, a single flow table and multiple flow table.

We used iPerf to generate UDP packets and send them to the server from the client, and we fixed the bandwidth at 100 Mbps. In this experiment, different payload sizes were used to evaluate the performance of the flow tables. As shown in Fig. 9, the throughput values indicate that the performance for large packets such as 1024 bytes and 1470 bytes was higher than that for small packets.

Because the NAT service is required to set the packet header field and a small payload tends to result in more packets being sent to the server simultaneously, the performance for larger packets is high. The throughput value of the multiple table framework is close to that of the single table framework. Thus, the multiple table framework is more flexible than the single table framework; however, the performance of the frameworks is similar (Fig. 9).

B. Evaluation of QoS When Host Bandwidth Is Limited

We verify our function by downloading a file, because downloading is a situation that always consumes network resources in practice. We downloaded an image of Ubuntu 14.04 that was approximately 1 GB in size.

The NFV controller ran on the Dell PowerEdge R630 rack server and executed QoS. The Edge-Core AS5712-54X [13] switch was used as the OpenFlow-enabled switch with the PicOS TM r2.6 operating system. We used a desktop computer as the experimental host to record the bandwidth every 2 seconds.

As shown in Fig. 10, we started downloading the file without rate limiting and the rate was between 40,000 and 70,000 Kbps. At the 8^{th} second, we limited the host 1 to 2048 Kbps and host 2 to 1024 Kbps by adding the rule of their MAC address. Then the bandwidth from host 1 and host 2 is decreasing. Host 1 and host 2 were limited to approximately 2048 Kbps and 1024 Kbps, respectively. At the 42^{nd} second, we change the bandwidth to host 2. That is, we set the bandwidth to host 2 from 1024 Kbps to 5120 Kbps. We observed that the bandwidth of the host 2 rapidly increased to 5120 Kbps.

C. Evaluation of QoS When Application Bandwidth Is Limited

In our experimental environment, we mirror all traffic from the SDN switch to application identification system and

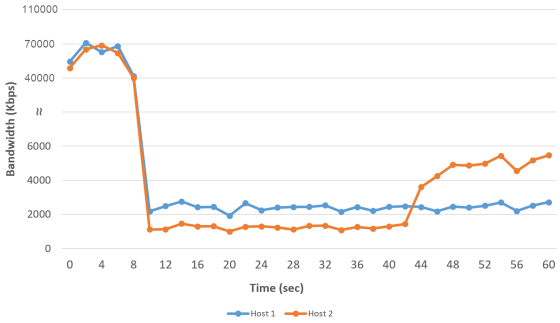

Fig. 10. Limiting the download rate of a host.

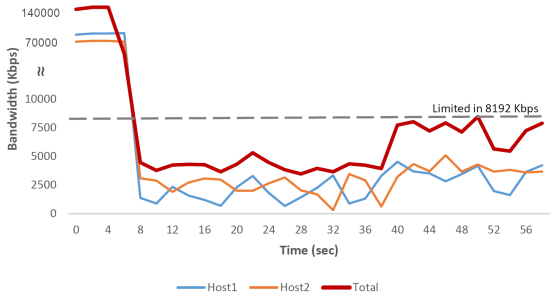

Fig. 11. Limit the rate of Filezilla in a network domain.

identify those flows belong to which application. Then the application identification system uploaded the identification results to the database. Subsequently, the controller received the results from the database and updated the flow information of the applications. Therefore, we could use the flow information (5-tuple and identification results) to limit the application bandwidth.

We selected Filezilla to verify our function for limiting applications and there are two hosts (host 1 and host 2) in the same network domain. Then we limited the bandwidth of Filezilla in this network domain. That is, the sum of bandwidths from host 1 and host 2.

As shown in Fig. 11. Initially, without limitation, the total bandwidth of Filezilla was approximately 140000 Kbps. At the 6th second, we limited the total bandwidth from Filezilla to 4096 Kbps. Then we observed that the total traffic from Filezilla in this network domain immediately decreased. Clearly, the total traffic from Filezilla was approximately 4096 Kbps. At the 38th second, we change the limitation from 4096 Kbps to 8192 Kbps and the total traffic from Filezilla increase to 8192 Kbps as expected.

Using this function for limiting applications, we can guarantee that the traffic in a network domain does not exceed the network capacity, thus preventing traffic congestion.

VI. Conclusion and Future Works

The proposed vCPE framework enables deploying NFVs as edge of network and these VNFs are implemented with multiple flow table management model. In this way, the customers only need a SDN switch at local network, and some network functions that could not be realized by single table mechanism are also resolved. The experimental results show that the framework provides better performance in VNF compete over single table SDN application. The integration evaluation also demonstrates its flexibility to integrate with any other bonus application identification system, IDS or IPS

This paper implemented multiple VNF cascade system over the single-table sdn switch. However, the multi-table environment in experiment basing on the single-table switch environment can't achieve the better performance compare with single-table switch. Although the multi-table based sdn switches are available on the market, those switches have different supporting table for different flow entry which might not be compatible to the customer requirement .If the algorithm is not appropriate designed for the specific network application entry, it will cause system crashed. We plan to design the algorithms specifically for multi-table switch in order to minimize the possibility of the crash of the system and maximize the performance.

References

[1] J. Matias, J. Garay, N. Toledo, J. Unzilla, and E. Jacob, "Toward an SDN-enabled NFV architecture," *IEEE Communications Magazine*, vol. 53, pp. 187–193, apr 2015.

[2] NFV, "Gs nfv 003 - v1.2.1 - network functions virtualisation (nfv); terminology for main concepts in nfv," tech. rep., dec 2014.

[3] NFV, "Gs nfv-inf 004 - v1.1.1 - network functions virtualisation (nfv); infrastructure; hypervisor domain," tech. rep., jan 2015.

[4] P. Minoves, O. Frendved, B. Peng, A. Mackarel, and D. Wilson, "Virtual CPE: Enhancing CPE's deployment and operations through virtualization," in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, Institute of Electrical and Electronics Engineers (IEEE), dec 2012.

[5] A. Lombardo, A. Manzalini, G. Schembra, G. Faraci, C. Rametta, and V. Riccobene, "An open framework to enable NetFATE (network functions at the edge)," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, Institute of Electrical and Electronics Engineers (IEEE), apr 2015.

[6] NFV, "Gs nfv-man 001 - v1.1.1 - network functions virtualisation (nfv); management and orchestration," tech. rep., dec 2014.

[7] "Ryu SDN framework." http://osrg.github.io/ryu/.

[8] M. Kuniar, P. Pereíni, and D. Kosti, "What you need to know about sdn flow tables," in *Passive and Active Measurement*, pp. 347–359, Springer Science + Business Media, 2015.

[9] O. N. Foundation, "The benefits of multiple flow tables and ttps," tech. rep., 2015.

[10] "Openflow switch specification." https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf, June 2012.

[11] "Pica8 OpenFlow-enabled switches." http://www.pica8.com/products/pre-loaded-switches.

[12] "iPerf." https://iperf.fr/iperf-doc.php.

[13] "Edge Core switches." http://www.edge-core.com/productsKind.php?cls=1.