Temp Title

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

***Abstract*—Traditional networks have been designed with purpose-built hardware network equipment and it took long cycles to develope and install these network elements. In this context, the concept of virtual Customer Premise Equipment (vCPE) is proposed to reduce OPEX and CAPEX. Software-defined networking (SDN) and network functions virtualization (NFV) are key roles for this innovation.**

In this paper, we proposed a vCPE framework enables deploying VNFs as edge of network. These VNFs are achieved by the synergies between an VNF controller on cloud and an SDN switch at the edge, and designed with our proposed multiple flow table management model. In this way, the customers only need a generic SDN switch at local network and they can obtain different network services by subscribing to different VNF controllers through our vCPE framework. We also perform two kinds of experiments: one is to evaluate the performance of VNFs implemented by the proposed management model, and the other is to demonstrates the flexibility to integrate with any other application identification system, IDS or IPS

***Index Terms*—SDN, NFV, vCPE, multiple flow tables**

I. Introduction

In recent times, researchers have shown an increasing interest in evolving network infrastructures. Software-defined networking (SDN) and network functions virtualization (NFV) are key roles for this evolution. SDN [1]–[4] has been widely studied for almost a decade since the first OpenFlow [5], [6] article had be presented in 2008. The main concept of SDN is separating data plane and control plane to enable smart control on switch and give a brand-new viewpoint on network research, and makes innovation on industries.

As SDN was developed, NFV [7]–[9] has been introduced by Telco operators at the same time. The network services offered by operators previously performed by specific hardware appliances and it is difficult to decrease the OPEX and CAPEX on service deployment and management. In this context, NFV is proposed to innovate in the service delivery arena. The concept of NFV is to reduce the coupling between network functions (NFs) and hardware devices. Virtual Customer Premise Equipment (vCPE) [10], [11], in particular virtual residential gateway (vRGW) [12], is one of the network services which benefited from NFV [13].

In the progress of vCPE development, SDN is not involved at first. Most of previous researches focused on other technology to virtualize and deploy the CPE node [14]–[19]. Cloud4NFV [20], [21], proposed by Portugal Telecom, started to use SDN technology on designing virtual CPE management and organization (MANO) platform for Telco cloud. Italy Telecom also proposed NetFATE [22], which is a network function deploy-to-edge model in which the NFs are designed by SDN and perform by SDN switch. Inspired from these two frameworks, we proposed a vCPE framework, attempting to replace hardware-based CPE. When the NFV is deployed at network edge and performed by SDN switch, there will be restriction on the OpenFlow Table [23]. In this paper, we also proposed a multiple OpenFlow table mechanism to implement network functions and explain how to use it to resolve the table restriction. We also evaluate the new VNF implemented by the proposed mechanism, and compare with the single-table mechanism. This new VNF can also be deployed to the network edge by the our vCPE framework.

II. vCPE System

A. Overview of Network Functions

Our network functions are designed with SDN-enabled NFV architecture concept [24], using the synergies between computing infrastructures (NFVs) and network infrastructures (NFVIs) [25], [26]. An NFV is a VNF controller, mainly used for addressing stateful processing and NFVI is an SDN switch used for stateless processing. 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 related studies that explored the virtualization of network function in PE devices [11] or used service-chains in data center to achieve vCPE services [27], we introduced a network function service deployment model based on the NetFATE (Network Function at the Edge) approach [22]; 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. 1 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 in which an SDN controller is run. The customer only needs to set up the gateway SDN switch to

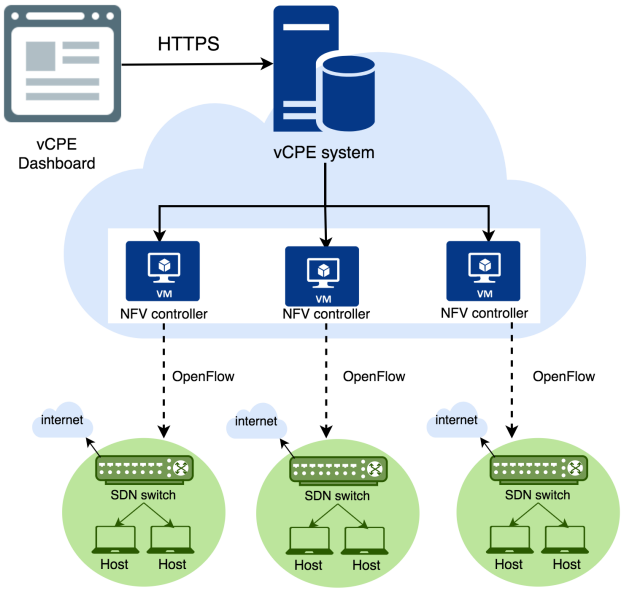

Fig. 1. Service deployment model.

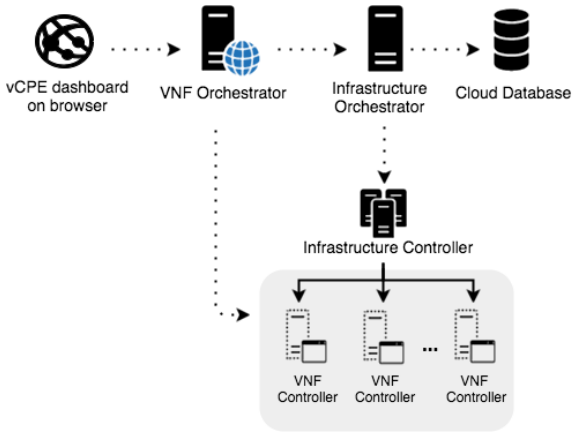

Fig. 2. Architecture of the vCPE system.

connect the SDN controller through the OpenFlow protocol; thereafter, the switch executes the service.

C. vCPE System Architecture

The architecture (Fig. 2) is adapted from the NFV-MANO architectural framework in [28], 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 and follows the request from the infrastructure orchestrator to create, delete, start, stop, and inspect containers.

2) Infrastructure Orchestrator: The infrastructure orchestrator automates the workflows of service deployment. When a customer subscribes to our service, the infrastructure orchestrator first authenticates the customer, calls the infrastructure controller to create a container for the customer, and then updates the information in the cloud database.

3) Cloud Database: The cloud database is used for restoring the meta data of our vCPE services, which include each customers credentials, customer's container/VM settings, and virtual CPE service states.

4) VNF Controllers: VNF controllers comprises an SDN controller developed using Ryu framework [29] and a remote launcher module. Once the infrastructure controller creates the container, the remote module will initially run, waiting for requests from VNF Orchestrator. The details of an SDN controller design are presented at Section III.

5) VNF Orchestrator: The VNF orchestrator provides web-based UI for vCPE services management and configuration. Through the VNF orchestrator, customers can subscribe to the desired service. 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.

III. Network Function With Multiple Flow Table Management Model

A. Multiple Flow Tables Strategy

In subsection II-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 [30] 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 [23], 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 resolve both restrictions, 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 [6]. The processing of each packet always starts at the first flow table. When being 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 execute the instruction set immediately, or execute after finishing the journey in switch. A flow entry can direct a packet to next table by go-to action.

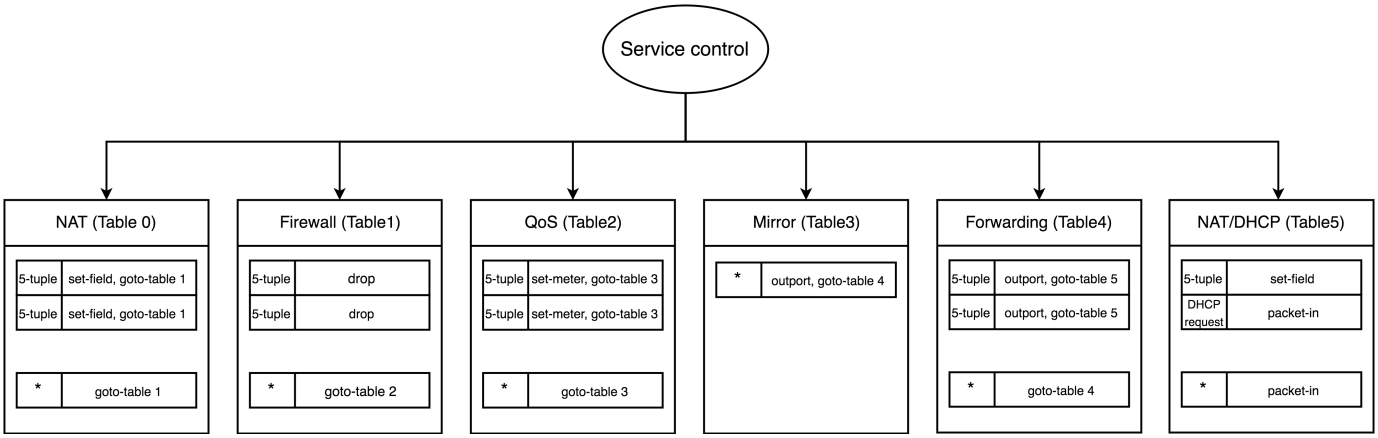

Fig. 3. Flow table order of vCPE service.

In our multiple flow table management model, we set the "go to next table" action as the table-miss action. Therefore, the packet is processed table-by-table in a certain sequence.

In a multiple flow table strategy, it is most important to determine which flow table the rules should be inserted into. We used the type of 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, NAT, DHCP, forwarding, traffic mirroring and QoS. The order of each function was determined as shown in Fig. 3 (note that the flow tables are counted from zero). 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 table", which enables the packets to continue to pass through all active services.

To disable a service, we merely add an enforce rule. Each enforce rule has the highest priority with the action, "go to next 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 based on MAC address, IP and layer4 ports. 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 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. Following is an example that illustrates the workflow of our NAT service. For an outgoing packet of a new connection, the SDN switch does not have any flow entry in the flow table, and hence, a packet-in event is triggered initially. In controller, the source IP and source port number of the outgoing packet are modified to a public IP address and a new port number is remapped for NAT service by Set-Field action. 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.

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. Our system can handle packets to realize the DHCP service.

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

The traffic mirroring service could make the manager to specify the output port to mirror the packet flow. It could make the network manager monitor the network situation easily. In our multiple flow table architecture, we use this service to mirror the packet flow to classifier which could identify the application. The QoS service could use the identified result to limit the application.

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.

H. QoS

Quality of Service (QoS) provides two strategies: hosts rate limitation and applications rate limitation.

1) Hosts Rate Limitation: 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.

2) Rate Limitation of Applications: In this strategy, we integrated a flow classification engine to identify the flow belongs to which application. To avoid some applications taking up a lot of bandwidth, we can limit the bandwidth for a certain application. We will mirror the packet to flow classification engine and identify which application belongs to. If we want to limit a specific application, we just add a 5-tuple rule and same meter to those packets which belong to this application. As a result, those applications will share the bandwidth by this meter.

3) The Flow Table Order of Forwarding and QoS Service: Because the order of NAT, DHCP, mirroring 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.

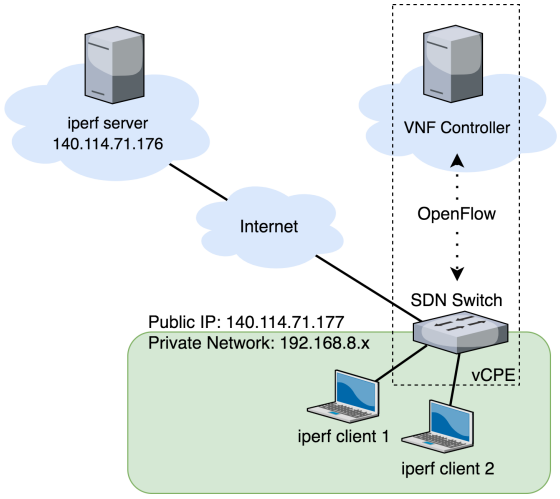

Fig. 4. Multiple Table Performance Evaluation Scenario

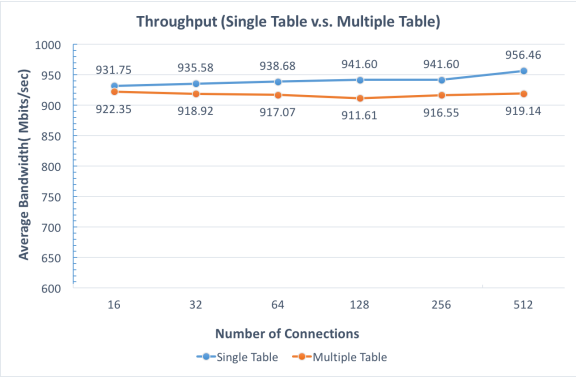

Fig. 5. Results of Multiple Table Performance Evaluation

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.

IV. 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. 4. The NFV controller was run on the Dell PowerEdge R630 rack server, and the EdgeCore AS5712-54X [31] was used as the SDN switch. We used iPerf2 [32] to generate network traffic. The iPerf server connected the public IP address with 140.114.71.176, 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 TCP packets and send them to the server from the client. In this experiment, different number of connections were used to evaluate the performance of the flow tables. As shown in Fig. 5, the throughput values indicate that the performance for the low number of connections are similar. But if we add number of connections, the performance of multiple flow table architecture is lower than single table architecture.

Because, at one connection the multiple table architecture add more rules than single table architecture. The throughput value of the multiple table framework is close to that of the single table framework at the low number of connections. Although the value of multiple table framework is lower than single table framework at the more number of connections, the multiple table framework is more flexible than the single table framework.

B. Evaluation of QoS When Host Bandwidth Is Limited

We verify our function by downloading a large 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 [31] 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. 6, we started downloading the file without rate limiting and the rate was between 40,000 and 70,000 Kbps. At the 8th 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 42nd 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

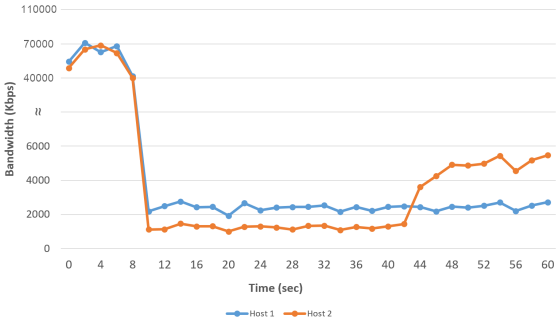

Fig. 6. Limiting the download rate of a host.

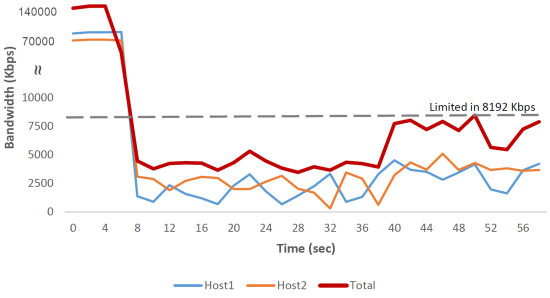

Fig. 7. Limit the rate of Filezilla in a network domain.

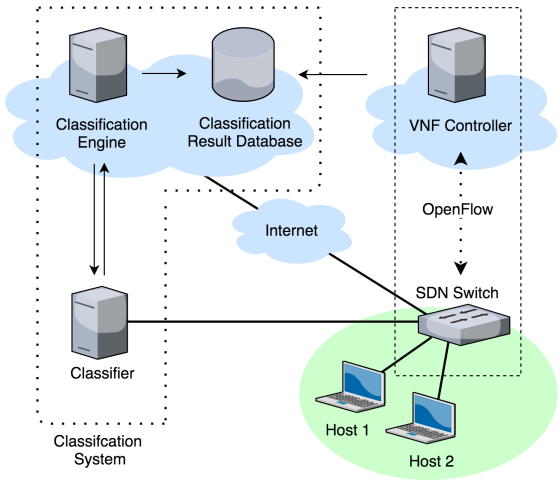

Fig. 8. Architecture of our application identification system in classification phase.

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. 7. 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.

V. 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] N. McKeown, "Software-defined networking," *INFOCOM keynote talk*, vol. 17, no. 2, pp. 30–32, 2009.

[2] O. N. Fundation, "Software-defined networking: The new norm for networks," *ONF White Paper*, vol. 2, pp. 2–6, 2012.

[3] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN," *ACM SIGCOMM Computer Communication Review*, vol. 44, pp. 87–98, Apr. 2014.

[4] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, pp. 14–76, Jan. 2015.

[5] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow," *ACM SIGCOMM Computer Communication Review*, vol. 38, p. 69, Mar. 2008.

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

[7] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng, *et al.*, "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action," in *SDN and OpenFlow World Congress*, pp. 22–24, 2012.

[8] NFV ISG, "Network Functions Virtualisation (NFV); Virtual Network Functions Architecture," Tech. Rep. GS NFV-SWA 001 V1.1.1, ETSI, Dec. 2014.

[9] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[10] NEC/Netcracker, "Nec's vcpe solution."

[11] 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*, IEEE, Dec. 2012.

[12] Z. Bronstein and E. Shraga, "NFV virtualisation of the home environment," in *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, IEEE, Jan. 2014.

[13] NFV ISG, "Network functions virtualisation (NFV); use cases," Tech. Rep. GS NFV 001 V1.1.1, ETSI, Oct. 2013.

[14] M. Ibanez, N. M. Madrid, and R. Seepold, "Virtualization of residential gateways," in *2007 Fifth Workshop on Intelligent Solutions in Embedded Systems*, IEEE, June 2007.

[15] M. Ibanez, N. M. Madrid, and R. Seepold, "Security management with virtual gateway platforms," in *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, IEEE, 2009.

[16] B. Zamaere, L. Da, and E. Kullberg, "On the design and implementation of a virtualized residential gateway," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, IEEE, Apr. 2012.

[17] N. Herbaut, D. Negru, G. Xilouris, and Y. Chen, "Migrating to a NFV-based home gateway: Introducing a surrogate vNF approach," in *2015 6th International Conference on the Network of the Future (NOF)*, IEEE, Sept. 2015.

[18] F. Sanchez and D. Brazewell, "Tethered linux CPE for IP service delivery," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, IEEE, Apr. 2015.

[19] R. Bonafiglia, S. Miano, S. Nuccio, F. Risso, and A. Sapio, "Enabling NFV services on resource-constrained CPEs," in *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*, IEEE, Oct. 2016.

[20] J. Soares, M. Dias, J. Carapinha, B. Parreira, and S. Sargento, "Cloud4nfv: A platform for virtual network functions," in *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, IEEE, Oct. 2014.

[21] J. Soares, C. Goncalves, B. Parreira, P. Tavares, J. Carapinha, J. P. Barraca, R. L. Aguiar, and S. Sargento, "Toward a telco cloud environment for service functions," *IEEE Communications Magazine*, vol. 53, pp. 98–106, Feb. 2015.

[22] 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)*, IEEE, Apr. 2015.

[23] Open Networking Fundation, "The benefits of multiple flow tables and ttps," tech. rep., ONF, 2015.

[24] 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.

[25] NFV ISG, "Network Functions Virtualisation (NFV); Terminology for Main Concepts in NFV," Tech. Rep. GS NFV 003 V1.2.1, ETSI, Dec. 2014.

[26] NFV ISG, "Network Functions Virtualisation (NFV); Infrastructure; Hypervisor Domain," Tech. Rep. GS NFV-INF 004 V1.1.1, ETSI, Jan. 2015.

[27] P. Cota and J. Sabec, "CPE virtualization by unifying NFV, SDN and cloud technologies," in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, IEEE, May 2016.

[28] NFV ISG, "Network Functions Virtualisation (NFV); Management and Orchestration," Tech. Rep. GS NFV-MAN 001 V1.1.1, ETSI, Dec. 2014.

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

[30] 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.

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

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