國立清華大學
NATIONAL TSING HUA UNIVERSITY

碩士論文

Enabling CPE virtualization with Multiple Flow Tables Architecture in SDN switches

利用軟體定義網路交換機之多流表架構實現網路設備虛擬化

所別　資訊工程研究所

學號　104062564

姓名　李紀萱 (Chi-Hsuan Li)

指導教授　黃能富博士 (Dr. Nen-Fu Huang)

中華民國一〇六年七月

Abstract

The virtual Customer Premise Equipment (vCPE) concept has been proposed recently to reduce OPEX and CAPEX. Software-defined networking (SDN) and network functions virtualization (NFV) are key roles for this innovation. This paper proposes a virtual network functions architecture with multiple flow tables. These VNFs are achieved by the synergies between a VNF controller on cloud and an SDN switch at the edge and deployed by the previous HSNL vCPE framework. The customer only needs a generic SDN switch at local network and it is very easy to subscribing different network services, such as Firewall, NAT, DHCP, and applications quality of service (QoS) by a browser-based dashboard. Experiments are conducted to evaluate the performance of VNFs implemented by the proposed multiple flow table management mechanism. The flexibility of architecture to integrate with other application classification system, such as IDS or IPS, is also demonstrated.

Keywords - Multiple flow tables, NFV, SDN, vCPE

中文摘要

近年，網路服務提供商爲了減少其資本性支出以及營運成本，提出網路設備虛擬化的概念，而軟體定義網路和網路功能虛擬化在此創新當中擔任關鍵的角色。本論文提出了一種多流表的虛擬網路功能架構，在此架構底下，這些虛擬化的網路功能通過在雲端的控制器和客户端的交換機之間的協同作用實現，還可以透過先前高速網路實驗室所提出的網路設備虛擬化之框架來部署，客户只需要擁有本地網絡上通用的軟體定義交換機，通過框架就可以輕鬆訂閱不同的網絡服務，如防火牆，網路地址轉換，動態主機設定協定和服務品質頻寬管理。實驗結果顯示，我們利用多流表管理機制實現之虛擬化網路功能的性能不亞於單流表之架構，更展示了與其他應用程序分類系統整合的靈活性。

Keywords - Multiple flow tables, NFV, SDN, vCPE

Contents

List of Figures

List of Tables

Chapter 1

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 [8–11] has been widely studied for almost a decade since the first OpenFlow [1, 12] article had be presented in 2008. SDN has been one of the pillars of innovation in network infrastructures, the main concept of SDN is separating data plane and control plane to enable smart control on switch and the programmability of the network through an open and standard interface[13]. This emerging network architecture give a brand-new viewpoint on network research and makes innovation on industries. Compared with the solutions on legacy network, it is more extensible on configuration and possible for enabling modifications of traffic flows on SDN architecture.

As SDN was developed, NFV [2, 3, 14] 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 to resolve these issues. The concept of NFV is to reduce the coupling between network functions (NFs) and specialized hardware devices and aims to replace traditional hardware-based network appliances with virtual appliances. Virtual Customer Premise Equipment (vCPE) [15, 16], in particular virtual residential gateway (vRGW) [17], is one of the network services which benefited from NFV [18].

The current issue with existing hardware CPE are difficult to provide value-added services, complex to introduce new services, and time-consuming and expensive for service deployment [15]. In contrast, vCPE abstracts the intelligence of such devices into software-based functionality and different services can act as different Virtual Network Functions (VNFs). Service providers will be able to provide services through Internet and the customer may need to buy only one low-cost device.

In the progress of vCPE development, the SDN is not involved at first. Most of previous researches focused on other technology to virtualize and deploy the CPE node [19–24]. Cloud4NFV [25, 26], proposed by Portugal Telecom, started to use SDN technology on designing virtual CPE management and organization (MANO) integrated cloud portal for Telco cloud. Italy Telecom also proposed NetFATE [4], 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, our laboratory, High Speed Network Laboratory (HSNL), also proposed a vCPE framework and a few network functions, attempting to replace hardware-based CPE [7, 27]. There are other CPE virtualization frameworks achieved by leveraging NFV and SDN. Ericsson proposed a Cloud-based vCPE solution [5] and Juniper also proposed a hybrid cloud CPE deployment model [6].

However, these SDN-involved vCPE research most focused on how SDN benefits

the design of NFV MANO [28, 29] platform or traffic steering between CPE nodes, not the CPE network function itself. NetFATE demonstrated an CPE architecture built with SDN technology, but the disadvantage of [4] is that it only tested the scenario of virtual firewall in this paper which lacked sufficient experimental results for other scenarios under NetFATE platform. [7] also focused on the the design of deployment framework and the most serious weakness is that the proposed services could not be enable at the same time since the network functions are designed in single flow table.

When the NFV is deployed at network edge and performed by SDN switch, there will be restriction on the OpenFlow Table [30]. In [30], 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. Each of the above categories can, in theory, be handled in a single table, but the handling is generally awkward and the need to combine matching fields forces an explosion of the number flow entries. Under these both restriction, the vCPE network function can not enable this To resolve both restrictions, we implemented the network functions by using a multiple flow table strategy.

Therefore, the purpose of this paper can be summarized as follows:

1. We proposed a multiple flow table mechanism to implement network functions which achieved vRGW and explain how to use it to resolve the table restriction.

2. The proposed CPE can enable forwarding, traffic mirroring, Network Address Translation (NAT), Dynamic Host Configuration Protocol (DHCP), firewall and QoS management to their customer at the same time.

3. To analysis the new VNF implemented by the proposed mechanism, this study

integrate all of functions mention above to evaluate it compare with the single-table mechanism.

This paper is structured as follows. Chapter 2 briefly introduces SDN technology, NFV architecture, the OpenFlow protocol, related studied of vCPE framework and the previous HSNL vCPE framework. In Chapter 3, we will review the NF design from the concept of SDN-enabled [31] architecture, and then move on describing our proposed multiple flow table mechanism. Chapter 4 turns to analyze the performance of vCPE network function what we proposed and compare to single table NF, followed by conclusion and future works in the last chapter.

Chapter 2

Related Work

In this chapter begins on a brief review of SDN in Section 2.1. Then we move on describing the detailed work of OpenFlow which is the first SDN standards and also the most well-known southbound protocol in Section 2.2. In Section 2.3, the NFV is introduced which is a network architecture concept of relocating network functions from dedicated appliances to generic servers.

Related works of virtual CPE platform will be described in Section 2.4. We will introduce NetFATE, which our HSNL vCPE framework is inspired from. We will also introduce vCPE products proposed by Ericsson and Juniper. Finally, we will described the HSNL vCPE framework in great detail in Section 2.5.

2.1 SDN

SDN has been one of the pillars of innovation in network infrastructures, allowing to decouple the control plane and data plane and enable the programmability of the network

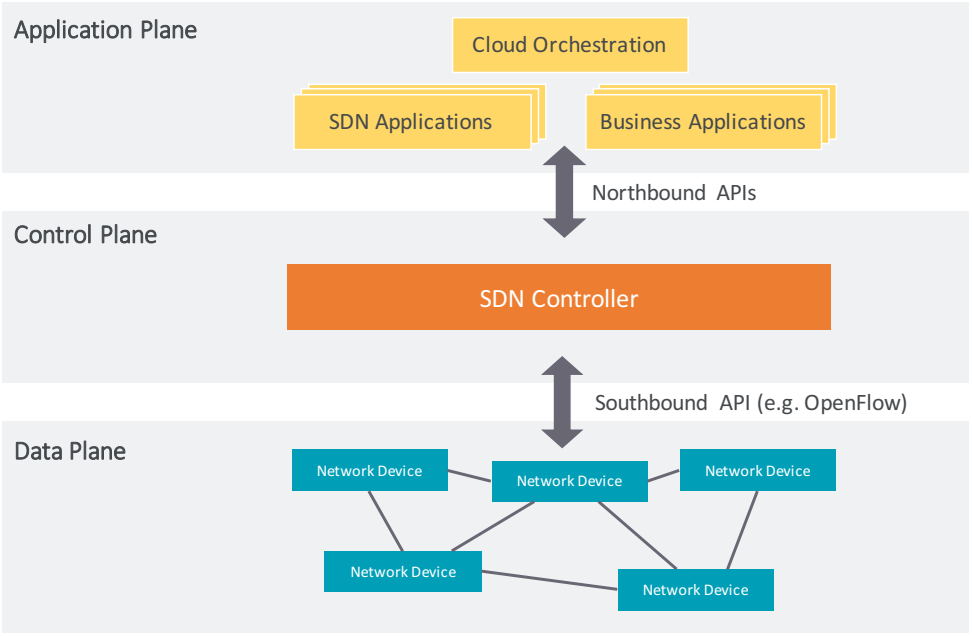


Figure 2.1: SDN logical architecture.

through an open and standard interface[13]. SDN also centralizes application controls of the whole network. With these benefits, network managers could directly implement their ideas on the network.

Figure 2.1 depicts a logical view of the SDN architecture. SDN applications and business applications are programs that built on top of the controller. These applications perform business logic and send requests to SDN controllers through northbound APIs. The most used northbound API is Representational State Transfer (REST) API.

When receiving the requests from applications, the SDN controller uses the southbound interface to communicate with programmable SDN network devices. The most commonly used is the OpenFlow specification [1] defined by Open Networking Foundation (ONF) [32], which will be presented in next section.

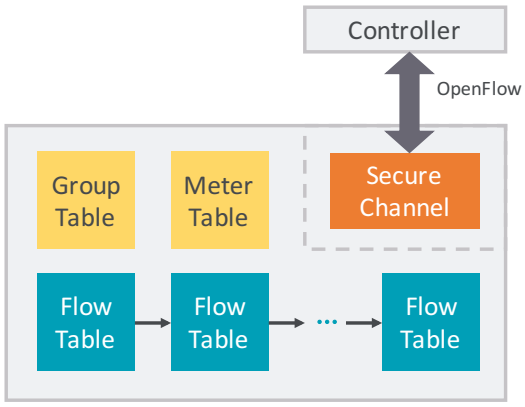


Figure 2.2: Main components of an OpenFlow switch. [1]

Match Fields	Priority	Counters	Instructions	Timeouts	Cookie

Figure 2.3: Main components of a flow entry in a flow table. [1]

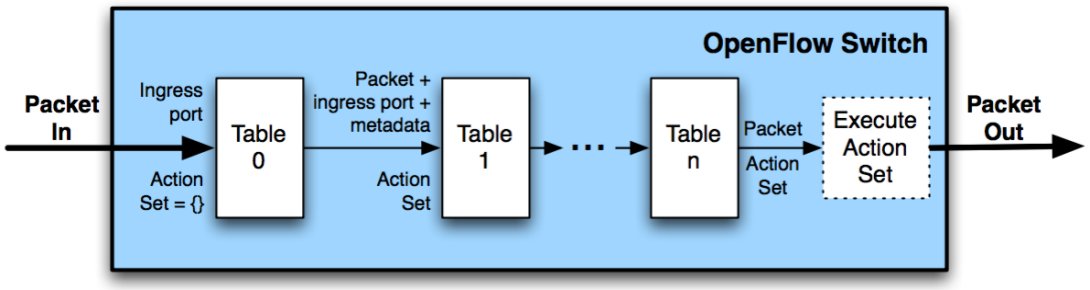
2.2 OpenFlow Protocol

2.2.1 OpenFlow Switch Components

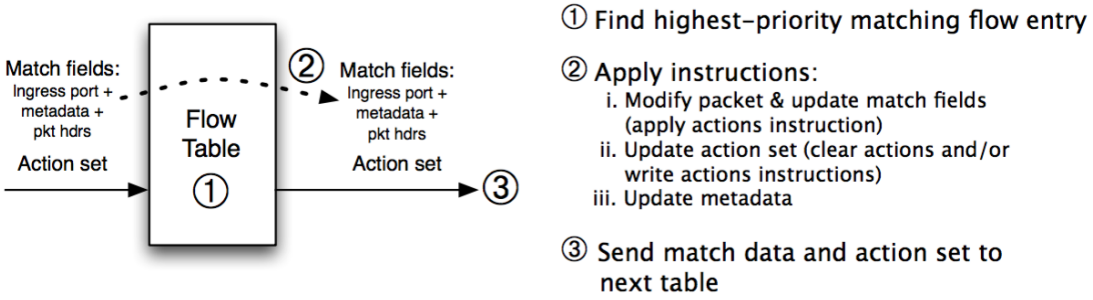
Fig. 2.2 illustrates basic architecture of the OpenFlow-compatible switch. In an OpenFlow Switch, there are multiple flow tables, a group table, a meter table and an OpenFlow channel.

There are flows entries in each flow table and each flow table entry contains match fields, priority, instructions, timeouts, and so on as shown in Fig. 2.3. The flow table matches incoming packets to a particular flow entry and specifies the actions that are needed to be performed on the packets.

The group table is used to arrange multiple flow entries into a certain group and update the actions based on the group. The meter table can trigger a variety of bandwidth



(a) Packets are matched against multiple tables in the pipeline.



(b) Per-table packet processing.

Figure 2.4: Packet flow through the processing pipeline. [1]

performance-related actions on a flow. Through the the OpenFlow channel, SDN controller can communicates to the switch.

2.2.2 Flow Table Pipeline, Matching and Table-miss

The Packet flow through the processing pipeline is described in Fig. 2.4 and the workflow of packet handling through an OpenFlow switch on receipt of a packet is illustrated in Fig. 2.5.

The processing of each packet always starts at the first flow table. When being processed by a flow table, the OpenFlow switch will perform a table lookup based on the

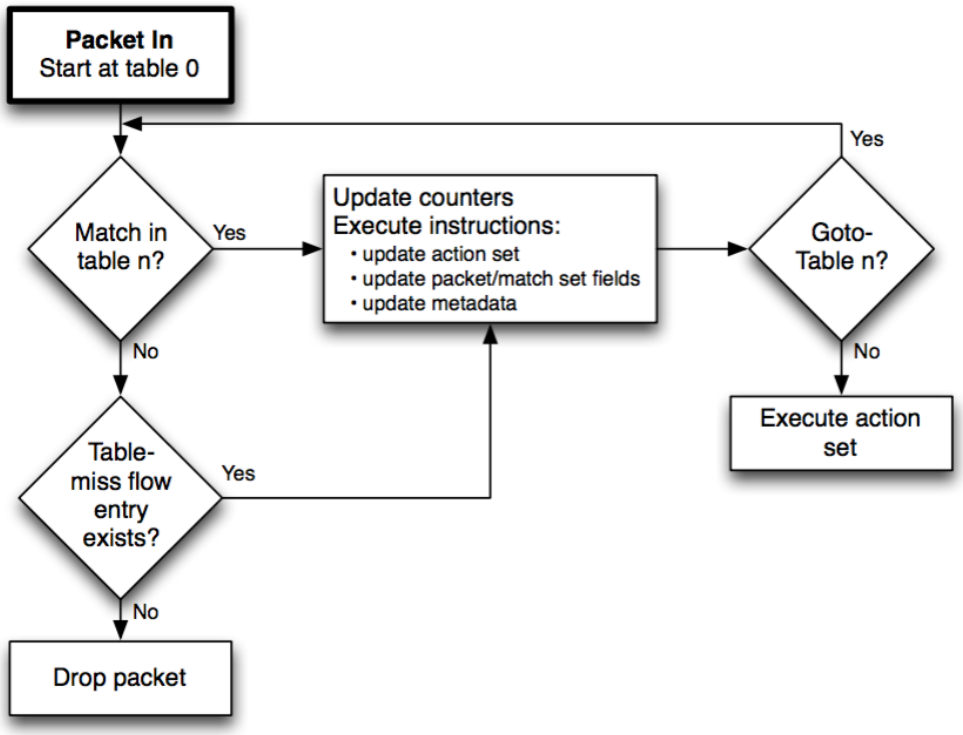


Figure 2.5: The workflow of packet handling through an OpenFlow switch. [1]

packet type and various packet header fields, such as source MAC address or destination IP address. If a flow table entry field has a value of ANY (field omitted), it matches all possible values in the header. Once the packet is matched highest-priority matching flow entry in the flow table, the counters on the selected flow entry must be updated and the corresponding action must be added to the instruction set.

The packet can execute the instruction set immediately (APPLY-ACTION option), or execute after finishing the journey in switch (WRITE-ACTION option). If a packet matches an entry containing SET-FIELD with APPLY-ACTION option in previous table, those changes are applied immediately and effecting the matching in the next flow table. A flow entry can direct a packet to next table if there's a GOTO-TABLE action in

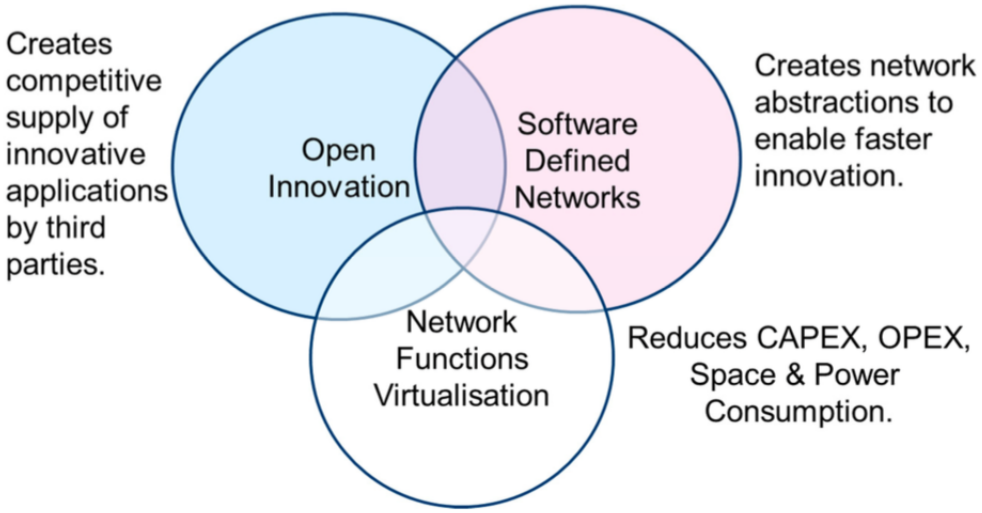


Figure 2.6: NFV Relationship with SDN. [2]

instruction set.

Every flow table must support a table-miss flow entry to handle the table missed condition, in which packets unmatched any flow entries in the flow table. The table-miss flow entry is identified by its wildcard patterns and the lowest priority (0). The instructions of the table-miss flow entry may be sending packets to the controller (PACKET-IN action), dropping packets (DROP action) or directing packets to a subsequent table (GOTO-TABLE action).

2.3 NFV

2.3.1 The concept of NFV

Legacy network functions developed by different vendors and provided based on different hardware. This makes them difficult to design, manage and deploy. To

resolve these issues, telecom providers came together in a group called European Telecommunications Standards Institute (ETSI) and the ETSI setups the Industry Specification Group for Network Functions Virtualization (ISG NFV). The working group proposed a new architecture for network virtualization paradigm. The original NFV white paper [2], described the problems that they are facing, along with their proposed solution.

NFV virtualizes network services via software and it decouples the network functions from specialized hardware and aims to replace traditional hardware-based network appliances with virtual appliances. The benefits of network functions virtualization are: (1) reducing CAPEX and OPEX, (2) encouraging more innovation and investment to bring new services quickly at much lower risk, (3) significant reduction in calls to customer support and (4) greater flexibility to scale up, scale down or evolve services

2.3.2 ETSI NFV MANO Model

Shown in Fig. 2.7, the right side of the ETSI-NFV architecture [3] are: NFV Orchestrator (NFVO), VNF Manager (VNFM), and Virtualized Infrastructure Manager (VIM). NFVO is responsible for the orchestration and management of NFVI resources and to implement network services on the NFVI. VNFM is responsible for the lifecycle management of VNF instances (instantiation, configuration, update, scale up/down, termination, etc.) VIM is responsible for controlling/managing the Network Function Virtualization Infrastructure (NFVI) resources.

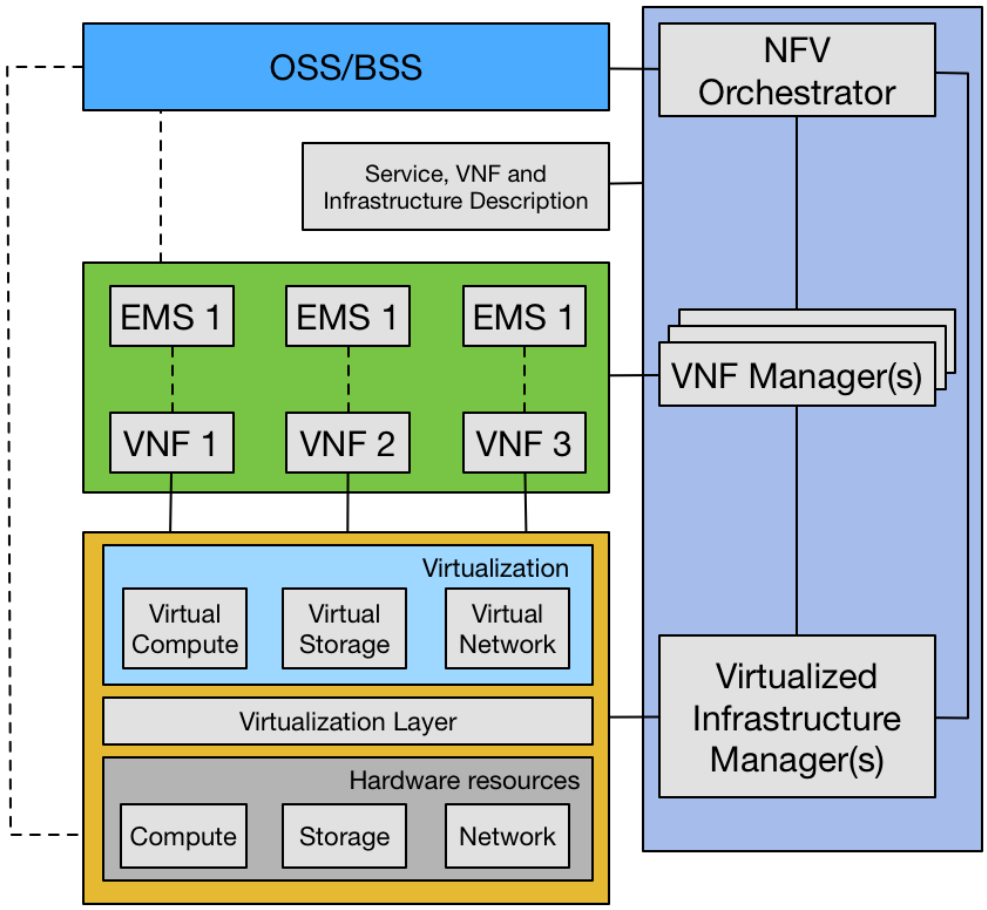


Figure 2.7: ETSI MANO Architecture. [3]

2.4 Related vCPE framework

2.4.1 NetFATE

NetFATE[4], proposed by Italy Telecom, which is a network function deploy-to-edge model in which the NFs are designed by SDN and perform by SDN switch. NetFATE architecture is compliant with the ETSI NFV architecture[3] and the so-called Virtual Network Function as a Service (VNFaaS) aims at virtualizing network functions of both

CPE devices and Provider Edge (PE) nodes.

The main components of NetFATE are CPE nodes, the PE nodes, the data centers, and the orchestrator. The CPE nodes are access gateways that can be either home gateways in a residential environment, or medium/high performance routers in an enterprise environment. The PE nodes are the aggregation nodes of a Telco network, and are typically shared by a high number of customers. Both CPE and PE nodes are included in the NFVI of NetFATE.

The orchestrator runs on a dedicated server and communicates with all the NFVI nodes through the Telco operator IP network. Its goal is to allocate, migrate and terminate VMs running network functions, and consequently controlling the traffic paths according to the run-time evolution of the network.

There is a case study as shown in Fig. 2.8 which is a virtual firewall based on the proposed NetFATE architecture. The personal virtual firewall services required by client 1 and client 2, respectively. When client 1 accesses the Internet through CPE 1 node, and then switches to the access point CPE 2 node without having any downtime. The orchestrator node in the NetFATE architecture plays a key role in the overall platform that migrates the personal virtual firewall from the previous CPE node to the new one.

The disadvantage of [4] is that it only tests the scenario of virtual firewall in this paper which lacks sufficient experimental results for other scenarios under NetFATE platform.

2.4.2 Ericsson CPE

Ericsson proposed a vCPE framework which achieved by unifying NFV, SDN and Cloud technologies in Internet providers' networks and data centers [5]. On the Fig. 2.9,

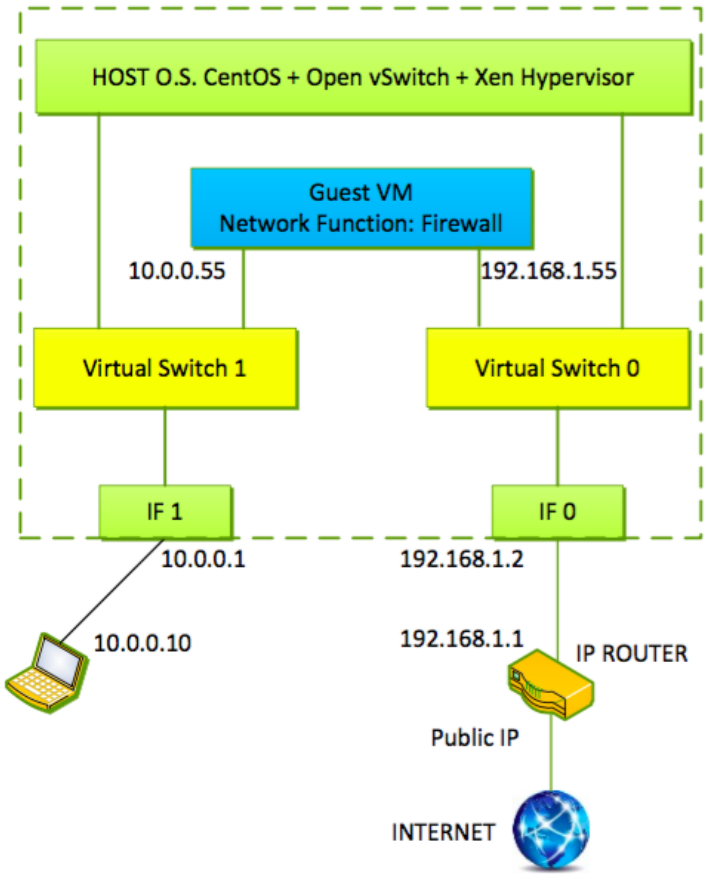


Figure 2.8: Virtual Firewall of NetFATE. [4]

it can be seen that Ericsson vCPE solution is built from several different components: access network, cloud data center, management and orchestration, applications (Virtual Network Functions) and self-care portal.

The access network is installed on the customer location and used to identify individual end-user device. There are 2 modes in access network: L2 tunneling mode and 2 bridge mode. These L2 mode configurations are both used to provide CPE solution to customer.

All services functions (SFs) can be hosted as virtualized instances in an SDN

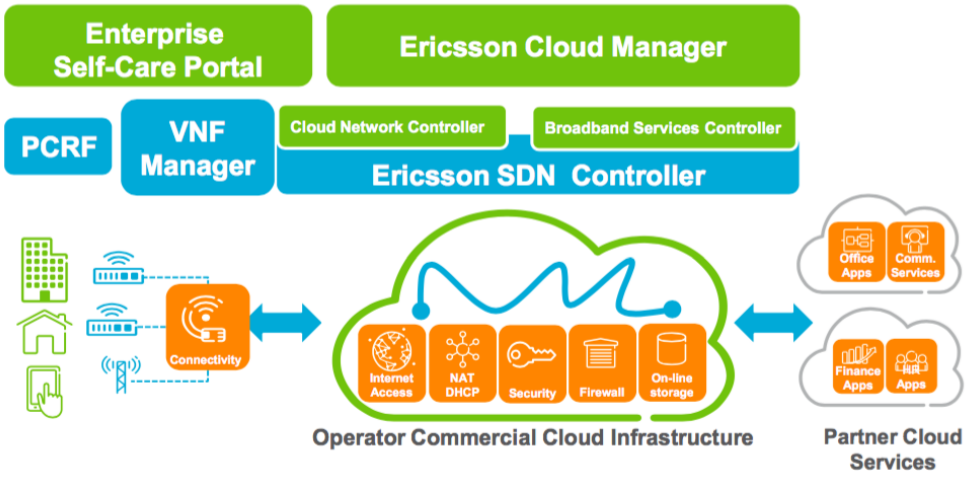


Figure 2.9: Ericsson vCPE architecture. [5]

enabled datacenter. Ericsson IaaS solution is called Cloud Execution Environment (CEE), modified from OpenStack. There are three roles in the cloud domain: Cloud SDN switch (CSS) accepts the configuration of OpenFlow command from the SDN controllers; Cloud SDN controller (CSC) centrally controls a network of CSS; Broadband Service controller (BBSC) configures an SDN network through CSC and handles traffic steering along the predefined service chain.

Ericsson Cloud Manager (ECM) is used for management and orchestration, which handles business logic and subscription services. It enables provisioning new service function instances, scaling and instantiating as requested on command. ECM also communicate with controllers in data center to send notifications about the deployed service functions and the provisioned service chains.

Self-care portal provides the main GUI to the enterprise or home administrators for executing different service request. Operator portal is provided towards cloud manager to transfer information to CEE and enterprise portal enables enterprise/home user to

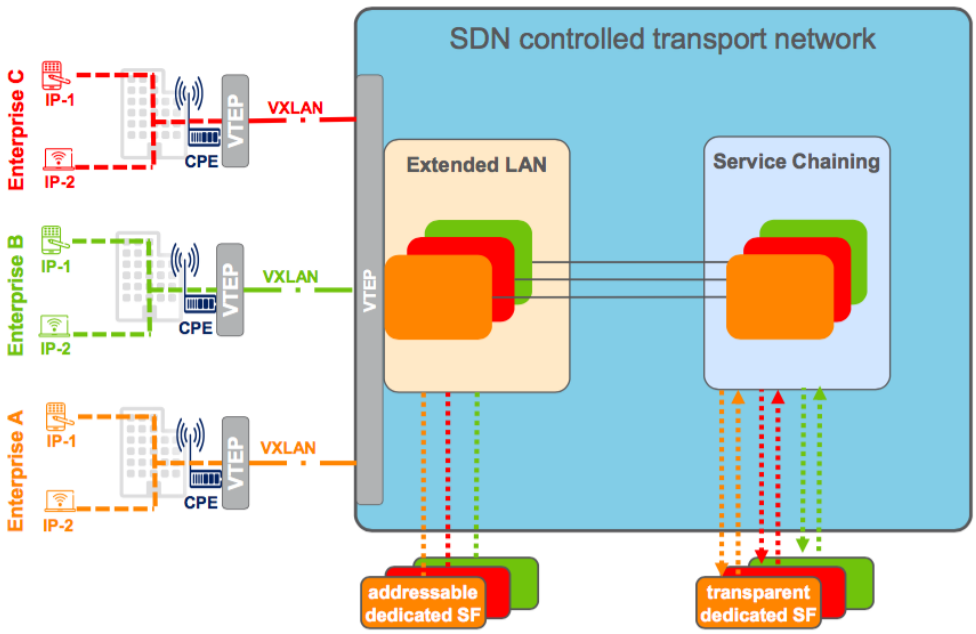


Figure 2.10: Overview of Ericsson Service Functions. [5]

monitor and act on the services available.

Then we move on to describing the SFs. There are two kinds of SFs: addressable dedicated SF and transparent dedicated SF, shown in Fig. 2.10. Addressable dedicated SF directly addresses end users with their MAC address. The traffic from end-used devices will be handle by SF and forward back immediately to the end-users without the processing of service chain. Transparent SFs use packet header to classify packet and identify data flow. The traffic from customers will be forward along the selected service chain.

The advantages of Ericsson vCPE solution is easy-to-deploy by using tunneling from customer to data center. However, the performance of the network functions may be a concern, since all traffic is needed to forward to data center.

2.4.3 Juniper Cloud CPE

Juniper proposed a Cloud CPE solution [6]. The solution have two kinds of deployment model: centralized cloud CPE deployment model and distributed cloud CPE deployment model.

In centralized cloud CPE deployment model, services can be ordered through a customer portal or triggered by existing business support system (BSS). BSS automatically instantiates VNFs and service chaining. The vCPE functions are achieved by these service chain. In distributed cloud CPE deployment model, they distributed Juniper NFX250 service platform to customer location, which provides similar functionality to the physical CPE. A single NFX 250 device can support multiple VNFs.

Juniper's Cloud CPE simultaneously can support both centralized and distributed model, called hybrid cloud CPE deployment model.

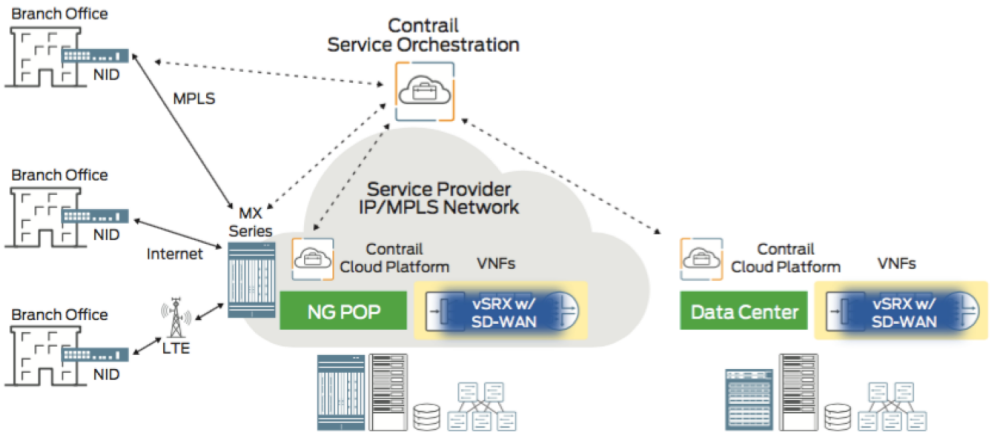
2.5 HSNL vCPE framework

HSNL virtual CPE platform [7, 27] is inspired by ETSI NFV MANO model and designed under the concept of the ETSI NFV reference architectural framework [3].

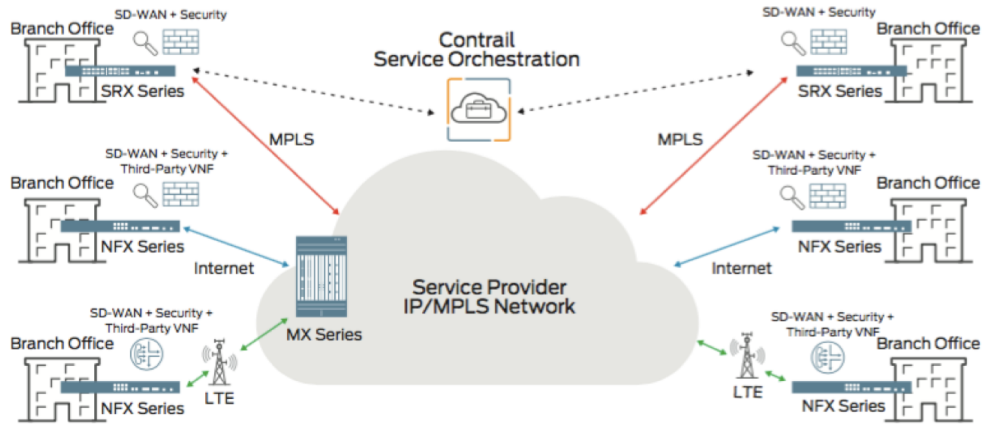
The following part of this paper moves on to describe in greater detail the HSNL vCPE framework. Our proposed vCPE functions, which is implemented by multiple flow table management mechanism, also can be deployed by this framework.

2.5.1 Deployment Model

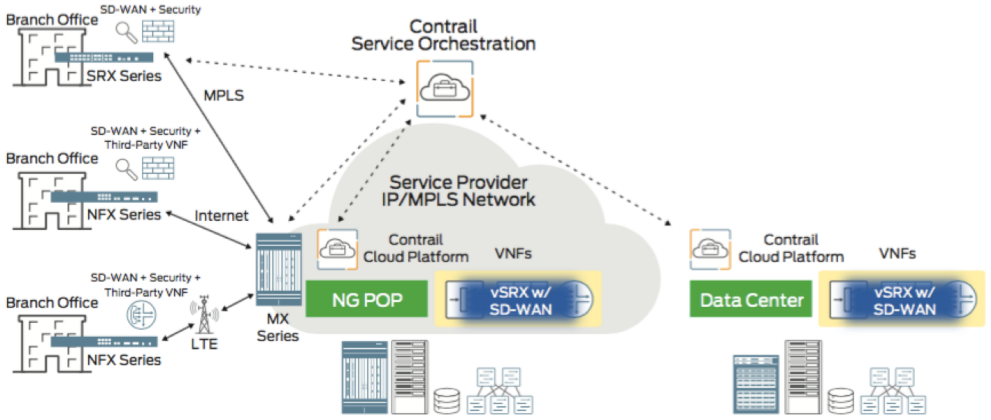
Unlike a related study that explored the virtualization of network function in PE devices [16] or used service-chains in data center to achieve vCPE services [5], HSNL



(a) Centralized deployment model.



(b) Distributed deployment model.



(c) Hybrid deployment model.

Figure 2.11: Juniper Cloud CPE deployment model. [6]

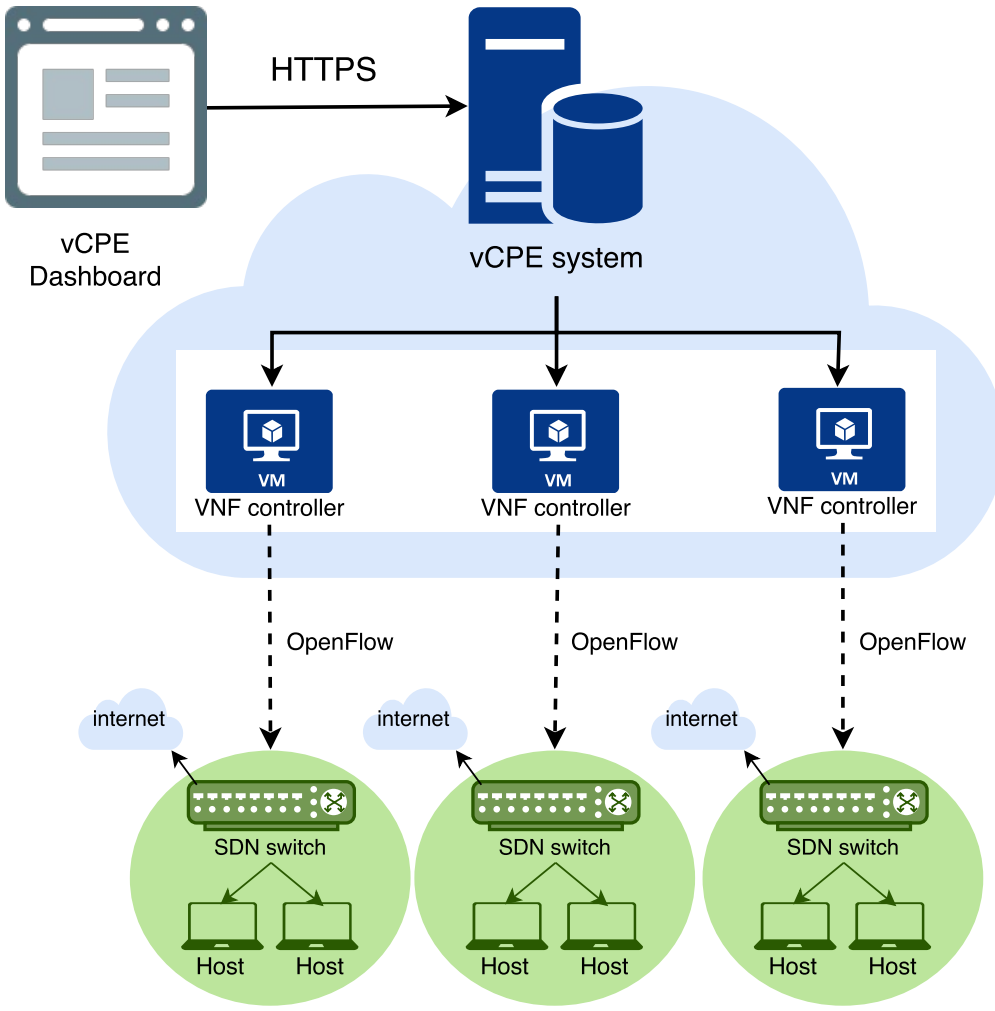


Figure 2.12: Service deployment model.

introduced a network function service deployment model based on the NetFATE (Network Function at the Edge) approach [4].

Fig. 2.12 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

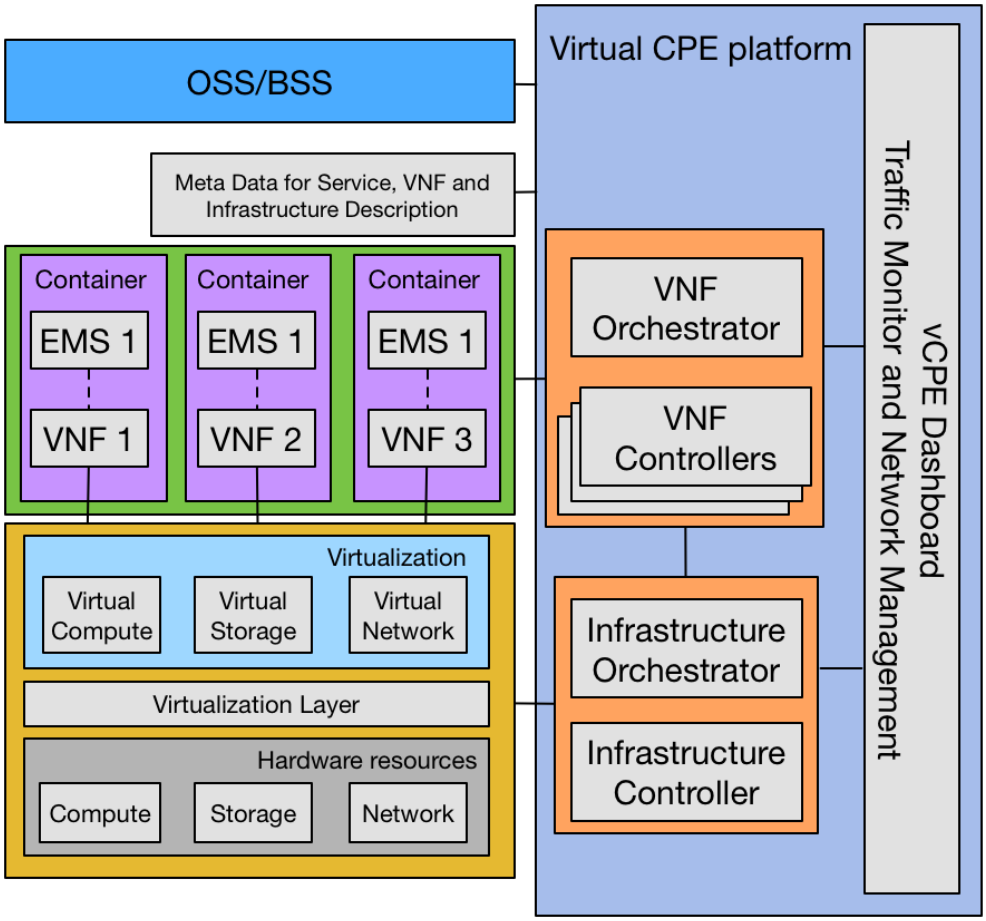


Figure 2.13: HSNL vCPE MANO Architecture [7].

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.

2.5.2 vCPE Platform NFV Architecture

The HSNL virtual CPE architecture (Fig. 2.13) expands the scope of ETSI NFV MANO model and we will go more detail in 2.5.3.

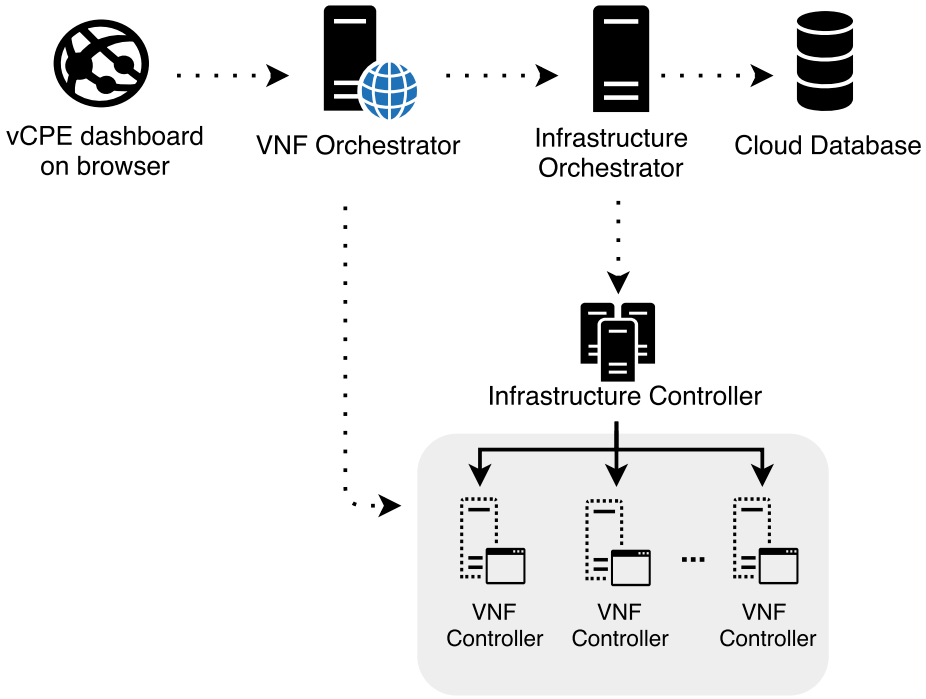


Figure 2.14: HSNL vCPE framework overview.

2.5.3 System Implementation

Turning now to the implementation of HSNL Virtual CPE platform. The system overview (Fig. 2.14) includes an infrastructure controller, an infrastructure orchestrator, a cloud database, VNF controllers and a VNF Orchestrator. Each component is introduced in the following parts.

2.5.3.1 Infrastructure Controller

The infrastructure controller comprises a Docker management server that can manage the Docker resources like containers and images. The infrastructure controller does not manage customer authentication or maintaining the state of the running service; it merely follows the request from the infrastructure orchestrator to create, delete, start, stop, and

inspect containers.

2.5.3.2 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 for the customer, and then updates the information in the database. The infrastructure orchestrator controls the entire life-cycle of our vCPE services.

2.5.3.3 Cloud Database

The cloud database is used for restoring the meta data of our vCPE services, which include each customer's credentials, customer's 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.

2.5.3.4 VNF Controllers

VNF controllers comprises an SDN controller developed using Ryu framework [33] and a remote launcher module. The original 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, the remote module will initially run,

waiting for requests from VNF Orchestrator. The virtual CPE functions are achieved by the synergies between the VNF controller and the SDN switch.

2.5.3.5 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, quality of service (QoS) management and our proposed virtual home gateway CPE functions which is implemented by multiple flow table mechanism.

Chapter 3

System Implementation

3.1 Overview of Network Functions

While [7] focused on the design of vCPE platform, this paper focused on the design of VNF itself. Our network functions (Fig. 3.1) are designed with SDN-enabled NFV architecture concept [31], using the synergies between computer infrastructures (NFVs) and network infrastructures (NFVIs) [34, 35]. An NFV is mainly used for addressing stateful processing and NFVI is used for stateless processing. In our architecture, we use an SDN controller as NFV and an SDN switch as NFVI.

3.1.1 Stateful Processing Component

This component is used to control the workflow of network function, 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.

3.1.2 Stateless Processing Component

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.

3.2 Multiple Flow Table Strategy

In section 3.1, we introduce the vCPE service design architecture. The network functions are achieved by the 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 them 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

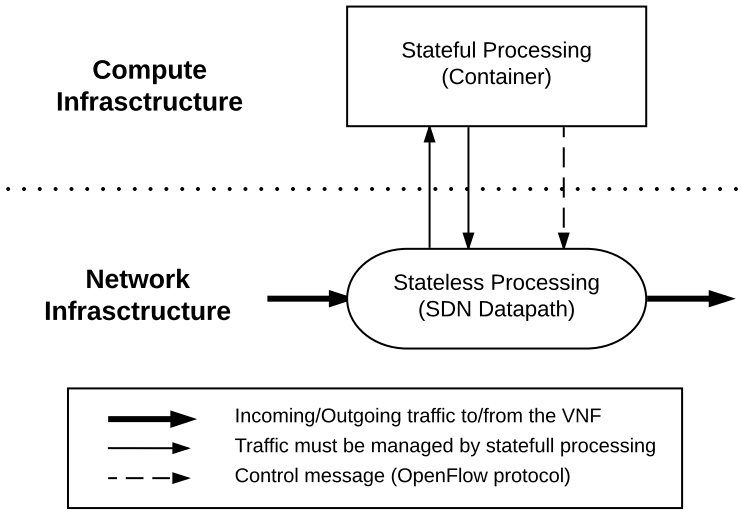


Figure 3.1: Overview of network functions.

the actions in matching rules. The flow table [36] 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 [30], 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.

The pipeline of OpenFlow flow table was illustrated in Section 2.2. In our multiple flow table management mechanism, we set the GOTO-TABLE N action as the table-miss action of table N-1. 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 purpose of network function as a demarcation,

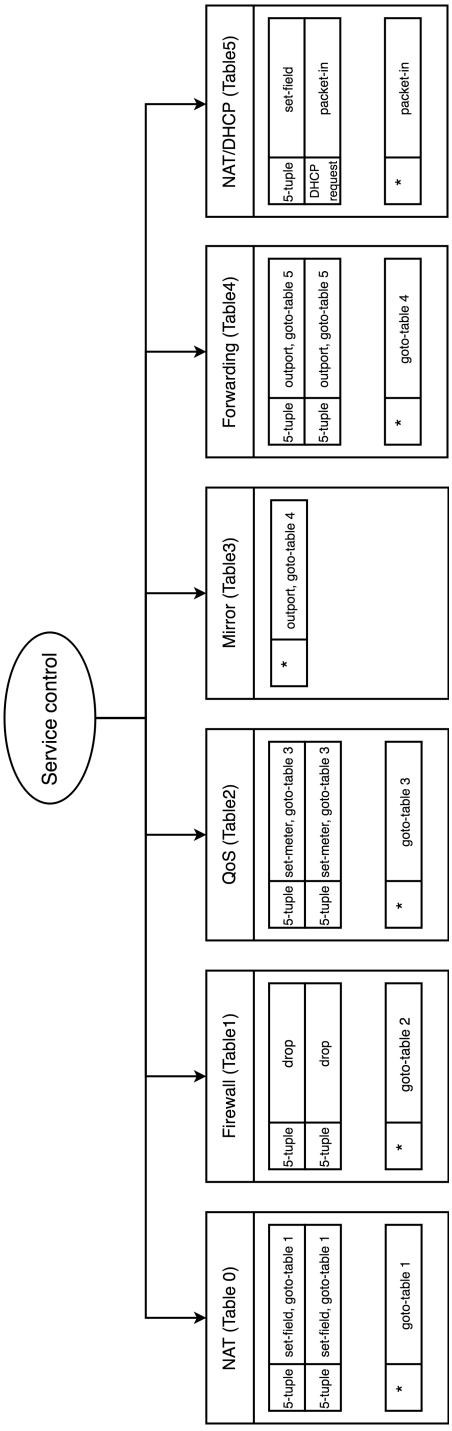


Figure 3.2: Flow table order of vCPE service.

that is, SDN applications responsible for specific purpose 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. 3.2 (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.

3.3 Service Control

Service control is used to enable or disable services. 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 GOTO-TABLE, which enables the packets to continue to pass through all active services.

To disable a service, a force-ignoring rule is added into the table of the service. The force-ignoring rule has maximum priority with the action GOTO-TABLE. To enable a service, we just remove the force-ignoring rule.

If the service we want to disabled is located at last table, we must not only modify the force-ignoring rule but also modify PACKET-IN rules.

3.4 Network functions

3.4.1 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 three kinds of policies:

1. block any traffic from a source IP or destination IP address;
2. block traffic based on known layer 4 protocols, such as SSH and HTTP;
3. block traffic with 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 3.1, 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 mechanism, 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.

Table 3.1: 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).

3.4.2 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 modification of the IP address and port number by using the NAT service. For a new connection, the first outgoing packet can't match any flow entry in SDN switch, so it will be sent to controller by the PACKET-IN action in the last table. The controller will modify the source IP to the public IP which is set by the network manager and modified the source port to an un-used port. Then, the controller will send the modified packet back to the switch. After that, the controller will send a pair of rule requests to SDN switch: one is for adding rules for egress traffic and the other is for ingress traffic.

There's an example shown in Fig. 3.3, the public IP address of NAT is 140.114.71.177, and the host private IP address is 192.168.8.254 with the port number 7777. The client sent the packet to a server with the IP address 140.114.71.178 and port number 8080.

As shown in Table 3.2, 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, 140.114.71.177, and the source port to 8888. When the server sends the packet back to the client (incoming),

Table 3.2: Flow entry for modifying the packet header fields

Direction	Egress	Ingress
Table	5	0
Match src IP	192.168.8.254	140.114.71.178
Match dst IP	140.114.71.178	140.114.71.177
Match src port	7777	8080
Match dst port	8080	8888
Action	src IP $\rightarrow$ 140.114.71.177; src port $\rightarrow$ 8888	dst IP $\rightarrow$ 192.168.8.254; dst port $\rightarrow$ 7777

the packet header field is modified. The destination IP address and destination port number are modified to 192.168.8.254 and 7777, respectively.

In the single flow table mechanism, the rules for NAT traffic contains not only SET-FIELD action but also the OUR-PORT action. We separate the SET-FIELD and OUT-PORT actions into two service: NAT and forwarding. The benefit of separating two kinds of rules is that the customers can decide enabling or disabling NAT service by themselves. If they disable the NAT service, they're still able to connect to the Internet by forwarding service. The previous NAT service designed by single table mechanism didn't achieve this flexibility. In the multiple flow table mechanism, the rules for handling ingress traffic are in the first table and rules for egress traffic are in the last table, since the QoS and forwarding service handle packets with private network packet header when NAT service is enabled.

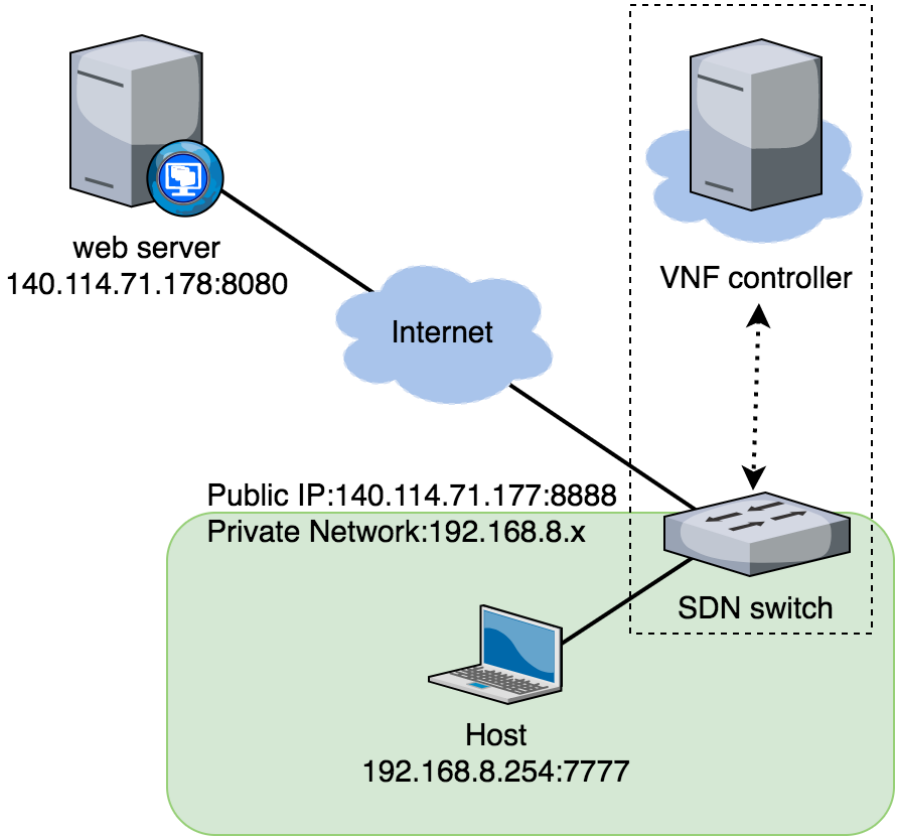


Figure 3.3: Example of modification of IP address and port number by NAT service.

3.4.3 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. If the incoming packet is a DHCP discovery packet, the controller assigns an IP address, generates a DHCP offer, and then performs a PACKEt-OUT event. If the incoming is a DHCP request, the controller generates a DHCP acknowledgement and PACKEt-OUT the generated packet.

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.

3.4.4 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.

The reason why we choose these 5 fields to form 5-tuple is to gather per-session statistical information. 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.

3.4.5 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 mechanism, we use this service to mirror the packet flow to classifier which could identify the application. The QoS service could use the classified result to limit the application.

3.4.6 QoS

QoS is mainly used for traffic control. Two management strategies are provided for QoS. We first introduce two strategies and then discuss the flow table order of QoS in the multiple table model.

3.4.6.1 Rate Limitation of Hosts

To implement the rate limiting for hosts, we use meter table to set the limitation in the specific bandwidth. With meters, we can create a meter for a desired bandwidth and assign the flows of target host with it.

3.4.6.2 Rate Limitation of Applications

In this strategy, we integrated a flow classification system to identify the flow belongs to which application. The integration scenario is presented in Fig. 3.4.

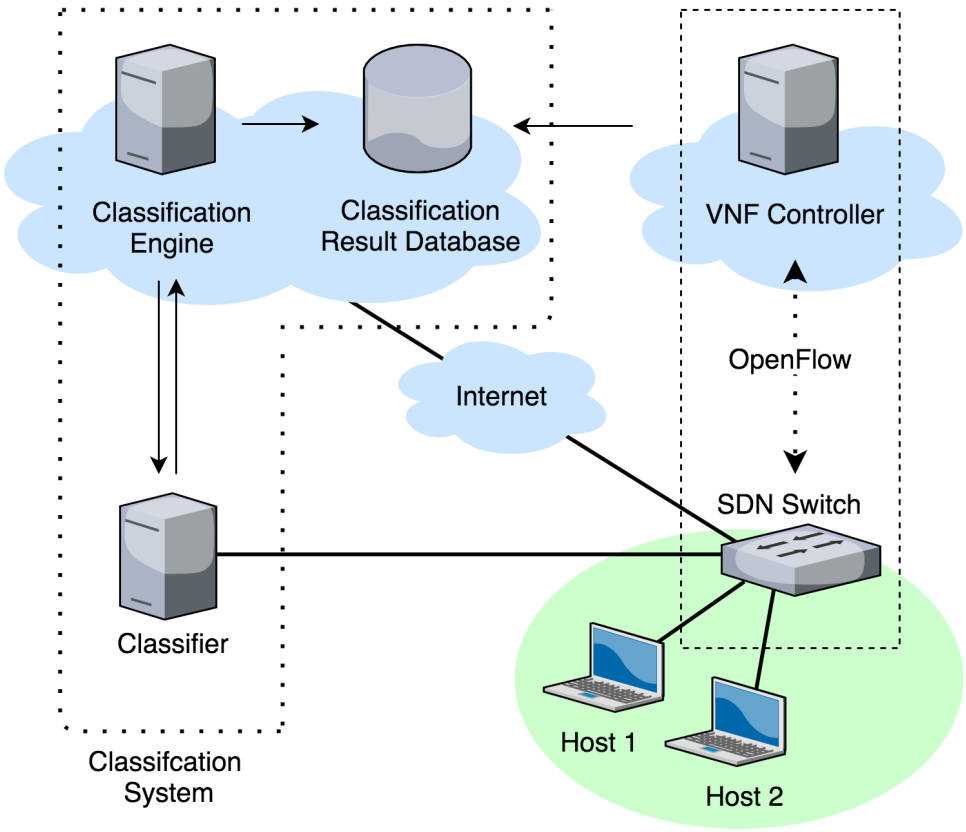


Figure 3.4: Scenario of integration our vCPE function with flow classification system.

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.

3.4.6.3 The Flow Table Order of Forwarding and QoS Service

Because the location of NAT, DHCP, firewall and mirroring 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. 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 controller's 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 controller's load decreases.

Chapter 4

Performance Evaluation

To evaluate the efficiency and flexibility of multiple flow table mechanism, the vCPE service with the proposed mechanism was deployed by the HSNL vCPE framework and two kinds of experiments are organized as follow: multiple table performance evaluation and integration evaluation. The performance measurement focused on measuring the NAT and forwarding performance on Virtual CPE. The integration evaluation was designed for proving the functionality of all network functions of our vCPE, and thus demonstrating the flexibility of our proposed multiple flow table mechanism to integrate with another services.

4.1 Multiple Table Performance

Implementing a single flow table application is easier than implementing a multiple flow table application; however, multiple flow table applications are more flexible. To verify the efficiency of the multiple flow table vCPE application, we conducted an

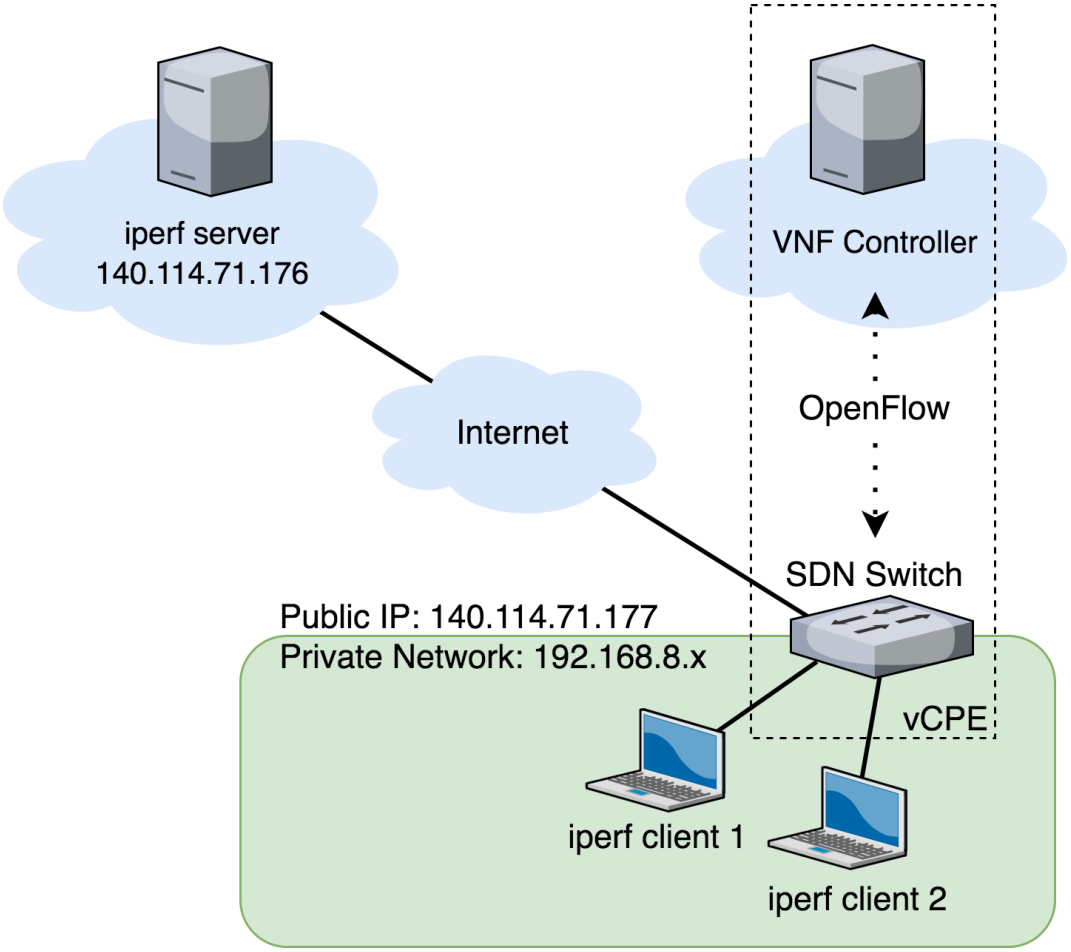


Figure 4.1: Multiple Table Performance Evaluation Scenario.

experiment by using the NAT and forwarding 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. 4.1. The NFV controller was run on the Dell PowerEdge R630 rack server, and the EdgeCore AS5712-54X [37] was used as the SDN switch. We used iPerf [38] 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

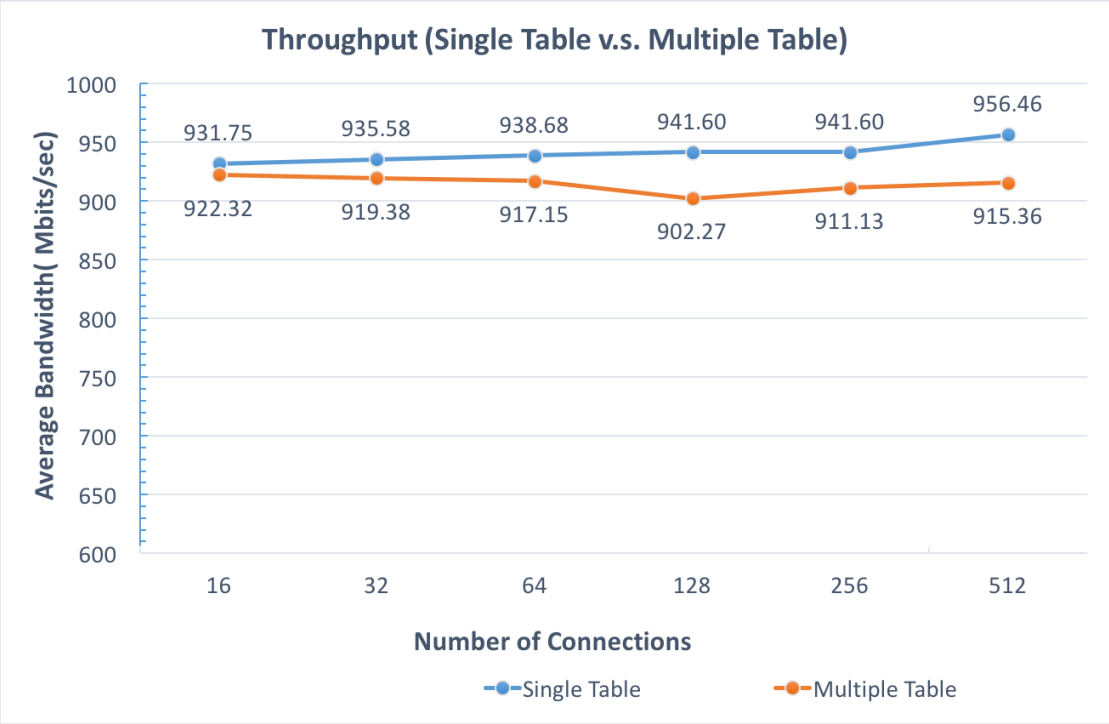


Figure 4.2: Results of Multiple Table Performance Evaluation.

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. 4.2, the throughput values indicated that the performance for the low number of connections are similar. But if we increased the number of connections, the difference of performance between the two applications performance become bigger and bigger.

For each connection, the multiple table application added 4 rules and single table application added merely 2 rules. Since the multiple table applications needed more rules than the other, the throughput of the multiple table framework is worse than the

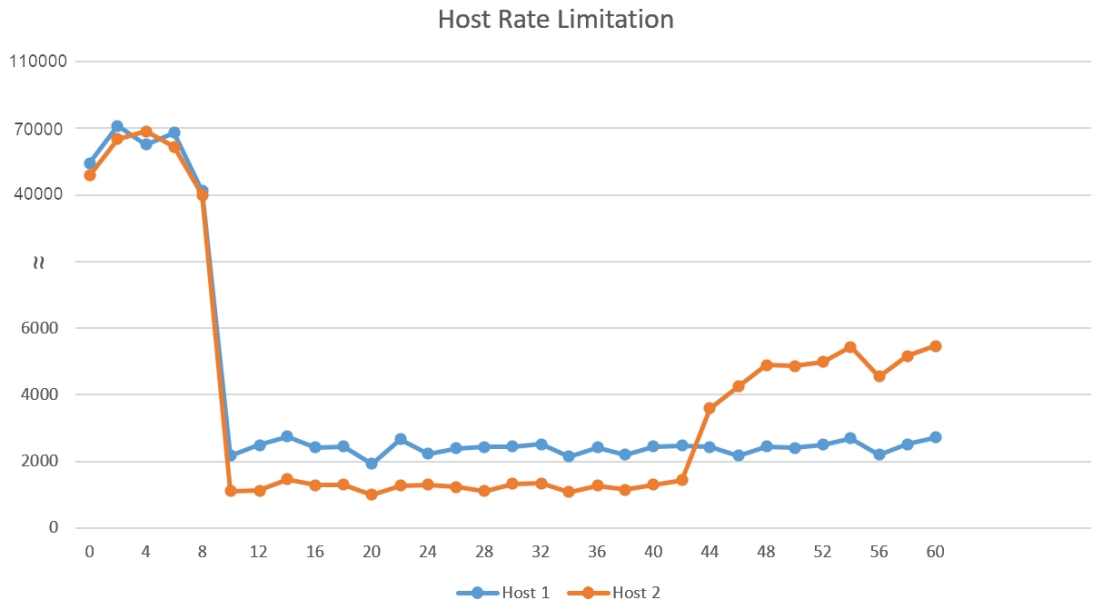


Figure 4.3: Limit the rate of a certain host.

throughput of the single table framework for the large number of connections conditions. 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 in add new services in to our vCPE.

4.2 Integration Evaluation

4.2.1 Evaluation of QoS When Host Bandwidth Is Limited

Because downloading is a situation that always consumes network resources in practice, we verify our function by downloading 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 [37] 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. 4.3, 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.

4.2.2 Evaluation of QoS When Application Bandwidth Is Limited

Our experimental environment is presented in Fig. 4.4. In our experimental environment, we mirror all traffic from the SDN switch to application classification system and identify those flows belong to which application. Then the application classification system uploaded the classification 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 classification 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. 4.5. Initially, without limitation, the total bandwidth of FileZilla

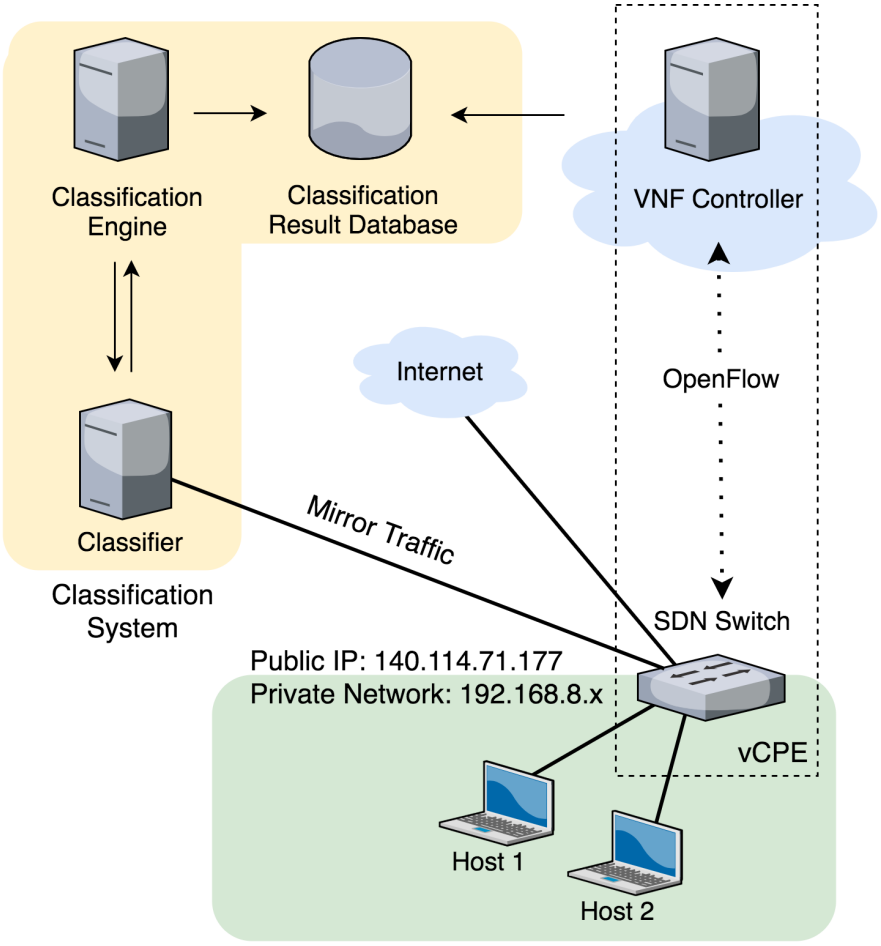


Figure 4.4: Scenario of integration evaluation.

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.

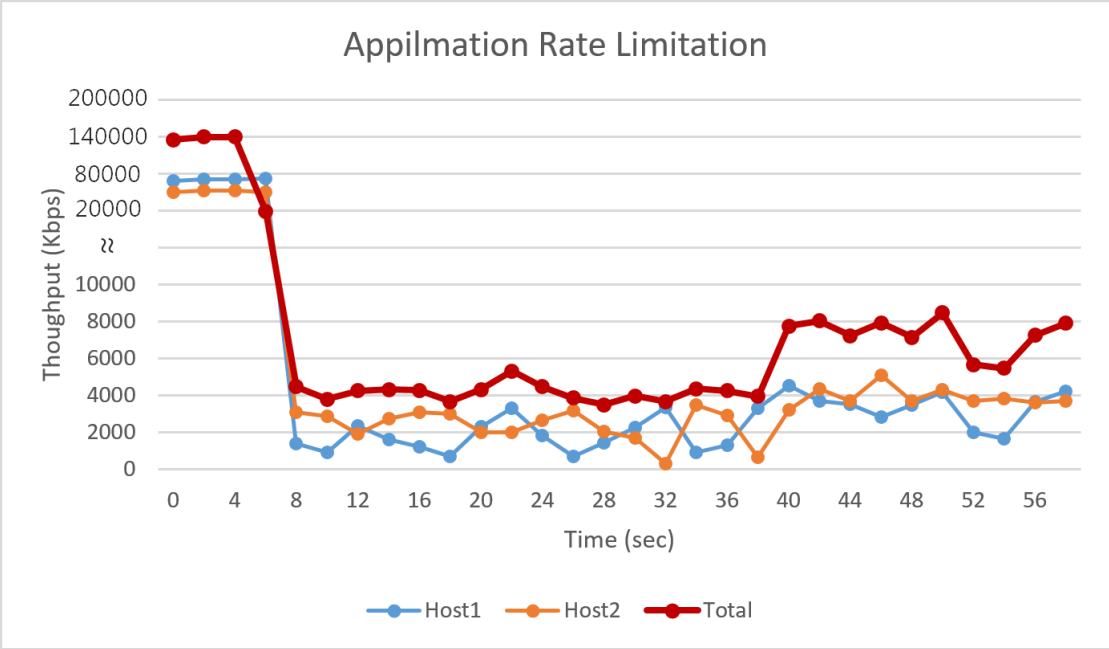


Figure 4.5: Limit the rate of FileZilla.

Chapter 5

Conclusion and Future Work

There has been a significant increase in using SDN technology to develop the vCPE. However, the studies are most focused on vCPE orchestration, deployment and management, not the vCPE functions. The purpose of this paper was therefore to discuss about the development of CPE function with the SDN technology. The proposed multiple flow table management mechanism was used to to implement the vRGW functions and unrestricted the single flow table limitation. The results of the experiment show that the new VNF provides provides same performance compare with single table SDN application.

Further research is needed, however, the multiple flow table mechanism need more flow rules in the SDN switch. The mechanism uses more space in flow tables to gain more functionality. Also, the order of network functions in flow table must be fixed and reduce the flexibility. As a future work, we plan to study further on the optimization of multiple flow table mechanism. We are still at an early stage of this approach and the full potential is yet to be revealed.

Bibliography

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

[2] 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*, 2012, pp. 22–24.

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

[4] 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.

[5] 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. [Online]. Available: https://doi.org/10.1109/mipro.2016.7522204

[6] "Juniper networks cloud CPE solution," https://www.juniper.net/assets/kr/kr/local/pdf/solutionbriefs/3510561-en.pdf.

[7] C.-W. Lin, "A Novel Virtual CPE Architecture and Service for Enterprises with SDN Network Technologies," Master's thesis, National Tsing Hua University, No.101, Sec. 2, Guangfu Rd., East Dist., Hsinchu City 300, Taiwan, 2016.

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

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

[10] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, Apr. 2014.

[11] 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, no. 1, pp. 14–76, Jan. 2015.

[12] 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, no. 2, p. 69, Mar. 2008.

[13] "Software-defined networking (SDN) definition," https://www.opennetworking.org/sdn-resources/sdn-definition.

[14] 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.

[15] NEC/Netcracker, "Nec's vcpe solution." [Online]. Available: http://www.nec.com/en/global/solutions/tcs/vcpe/

[16] 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.

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

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

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

[20] ——, "Security management with virtual gateway platforms," in *2009 Third International Conference on Emerging Security Information, Systems and Technologies*. IEEE, 2009.

[21] 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.

[22] 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, Sep. 2015.

[23] 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.

[24] 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.

[25] 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.

[26] 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, no. 2, pp. 98–106, Feb. 2015.

[27] N.-F. Huang, C.-W. Lin, S.-J. Wu, C.-H. Li, and I.-J. Liao, "A novel virtual cpe architecture and service for enterprises with sdn network technologies," in *PROCEEDINGS OF THE 9TH IEEE INTERNATIONAL CONFERENCE ON UBI-MEDIA COMPUTING" UMEDIA-2016"*, 2016, pp. 104–109.

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

[29] ——, “Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework,” ETSI, Tech. Rep. GS NFV-EVE 005 V1.1.1, Dec. 2015.

[30] Open Networking Foundation, “The benefits of multiple flow tables and ttps,” ONF, Tech. Rep., 2015.

[31] J. Matias, J. Garay, N. Toledo, J. Unzilla, and E. Jacob, “Toward an SDN-enabled NFV architecture,” *IEEE Communications Magazine*, vol. 53, no. 4, pp. 187–193, Apr. 2015.

[32] “Open network foundation,” https://www.opennetworking.org/.

[33] “Ryu SDN framework,” http://osrg.github.io/ryu/.

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

[35] ——, “Network Functions Virtualisation (NFV); Infrastructure; Hypervisor Domain,” ETSI, Tech. Rep. GS NFV-INF 004 V1.1.1, Jan. 2015.

[36] M. Kuźniar, P. Perešíni, and D. Kostić, “What you need to know about SDN flow tables,” in *Passive and Active Measurement*. Springer Science + Business Media, 2015, pp. 347–359.

[37] “Edge Core switches,” http://www.edge-core.com/productsKind.php?cls=1.

[38] “iPerf,” https://iperf.fr/iperf-doc.php.