國立清華大學

NATIONAL TSING HUA UNIVERSITY

碩士論文

Temporarily Title

中文標題

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

Acknowledgement

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 [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, the 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, our laboratory, High Speed Network Laboratory (HSNL), also proposed a vCPE framework and a few network functions, attempting to replace hardware-based CPE [23, 24].

However, these SDN-involved vCPE research most focused on how SDN benefits the design of NFV MANO [25, 26] platform or traffic steering between CPE nodes, not the CPE network function itself. When the NFV is deployed at network edge and performed by SDN switch, there will be restriction on the OpenFlow Table [27]. In this paper, we 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 previous HSNL vCPE framework.

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 [28] architecture, and then move on describing our proposed multiple flow table management mechanism, which achieved vRGW functions. Chapter 4 turns to analyze the performance of vCPE network function what we proposed and compare to single table NF and traditional network devices, followed by Conclusion and proposed future works in the last chapter.

Chapter 2

Related Work

2.1 SDN

2.2 OpenFlow

2.3 NFV

2.4 Related vCPE framework

2.4.1 Cloud4NFV

2.4.2 NetFate

2.4.3 Ericsson CPE

2.5 HSNL vCPE framework

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

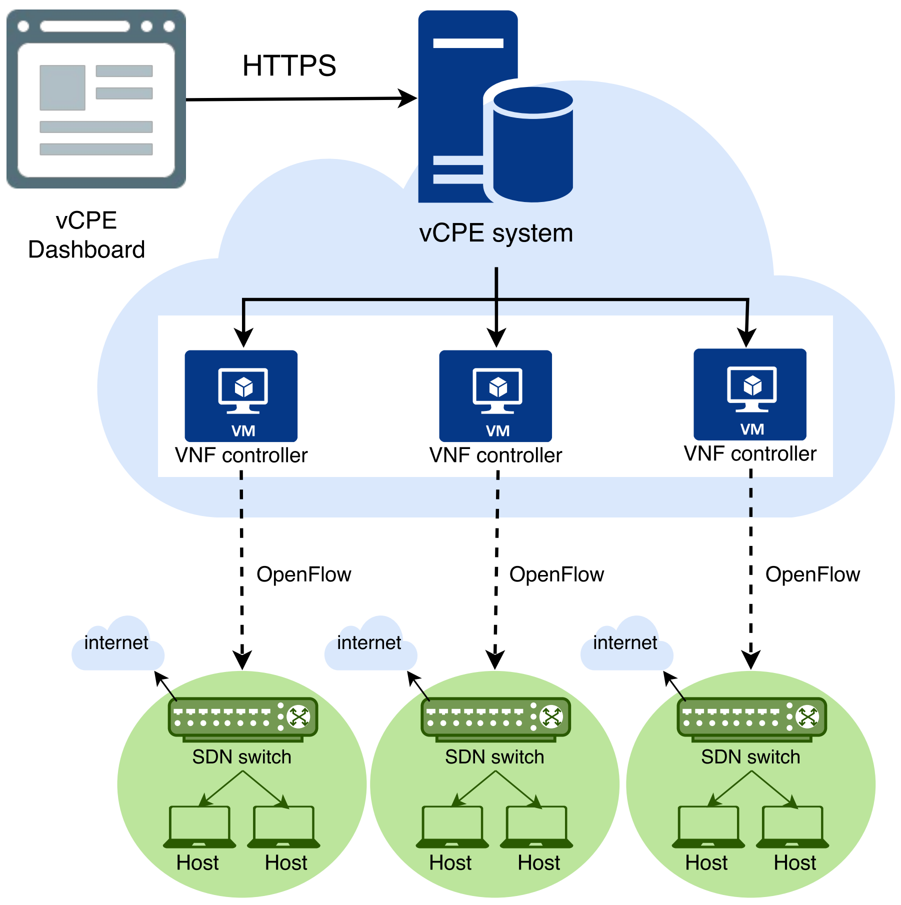


Figure 2.1: Service deployment model.

2.5.1 Deployment Model

Unlike a related study that explored the virtualization of network function in PE devices [11] or used service-chains in data center to achieve vCPE services [29], HSNL introduced a network function service deployment model based on the NetFATE (Network Function at the Edge) approach [22].

Fig. 2.1 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,

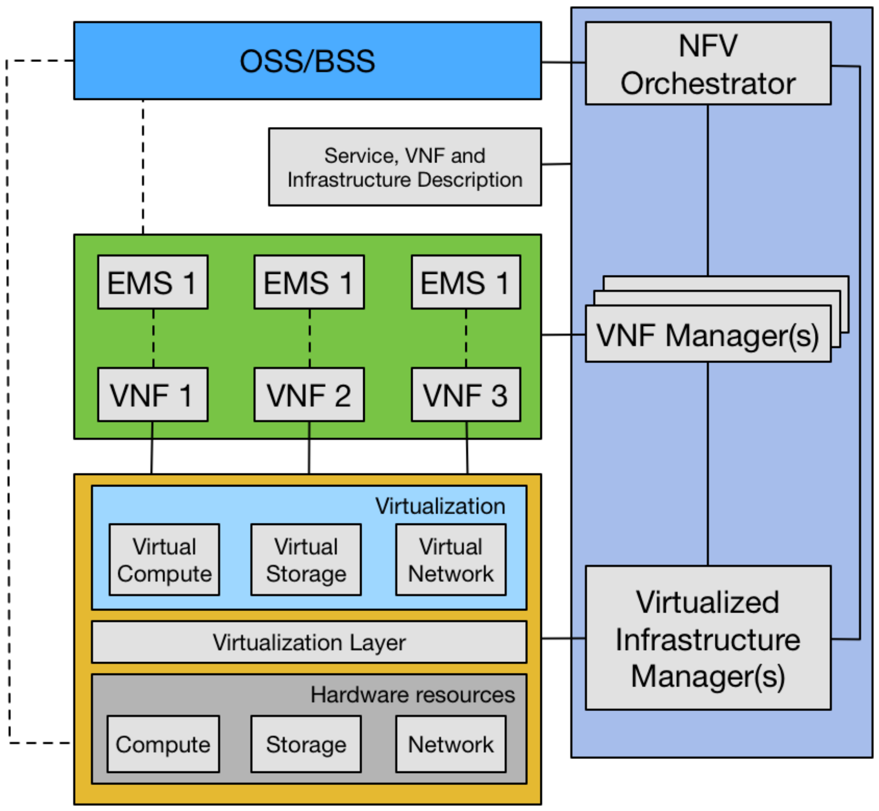


Figure 2.2: ETSI MANO Architecture.

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

2.5.2 Architecture of the Main System

2.5.2.1 Reference Architecture - ETSI NFV MANO Model

HSNL virtual CPE platform [23, 24] is inspired by ETSI NFV MANO model and designed under the concept of the ETSI NFV reference architectural framework [8].

Shown in Fig. 2.2, the right side of the ETSI-NFV architecture are: NFV Orchestrator (NFVO), VNF Manager (VNFM), and Virtualized Infrastructure Manager (VIM). NFVO

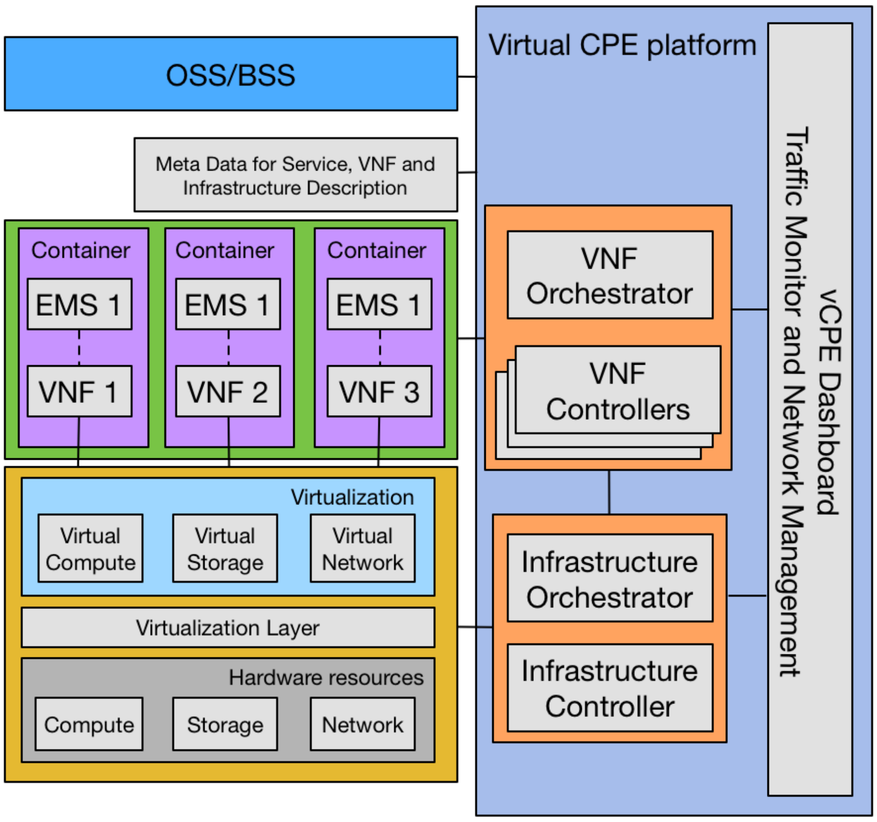


Figure 2.3: HSNL vCPE MANO Architecture.

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 NFVI resources.

2.5.2.2 vCPE Platform NFV Architecture

The HSNL virtual CPE architecture (Fig. 2.3) expands the scope of ETSI NFV MANO model and we will go more detail in 2.5.3.

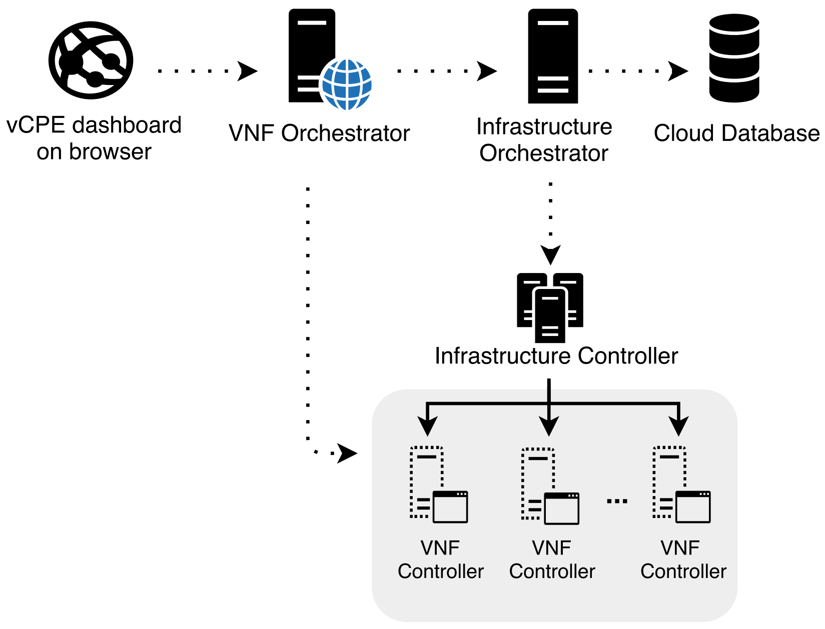

vCPE dashboard on browser
VNF Orchestrator
Infrastructure Orchestrator
Cloud Database
Infrastructure Controller
VNF Controller
VNF Controller
...
VNF Controller

Figure 2.4: HSNL vCPE framework overview.

2.5.3 System Implementation

Turning now to the implementation of HSNL Virtual CPE platform. The system overview (Fig. 2.4) includes an infrastructure controller, an infrastructure orchestrator, a cloud database, VNF controllers and a VNF Orchestrator. Each component is introduced in the following subsubsection.

2.5.3.1 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.

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 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.

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 [30] 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 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 mechnism.

Chapter 3

System Implementation

3.1 Overview of Network Functions

While [23] 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 [28], using the synergies between computer infrastructures (NFVs) and network infrastructures (NFVIs) [31, 32]. 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, 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.

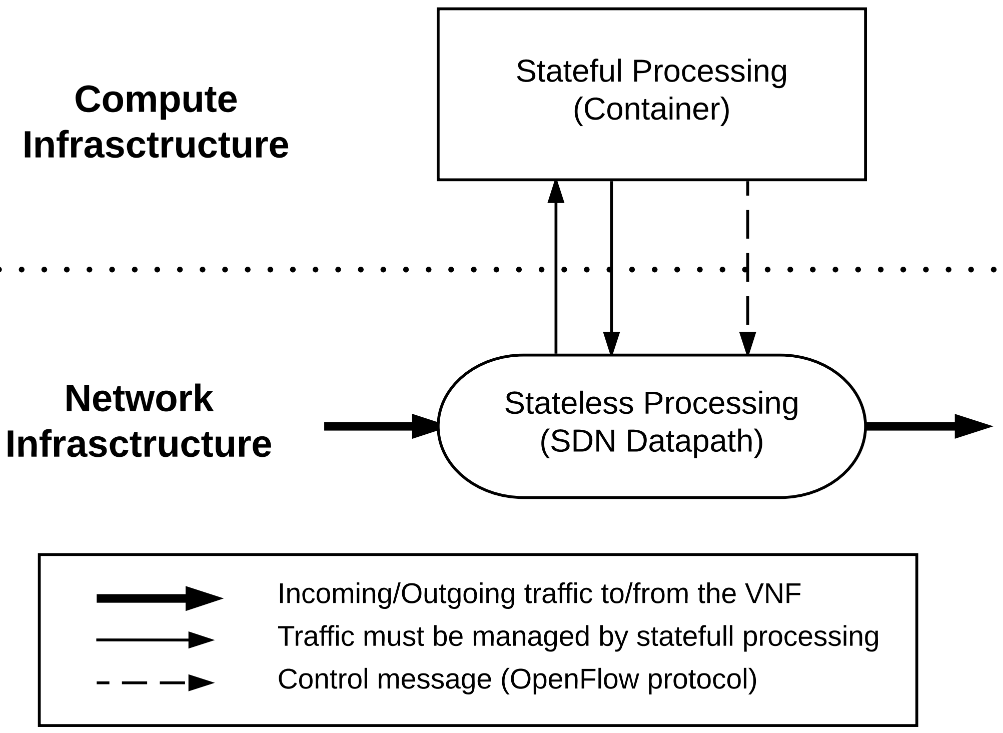

Compute Infrasctructure
Stateful Processing (Container)
Network Infrasctructure
Stateless Processing (SDN Datapath)
Incoming/Outgoing traffic to/from the VNF
Traffic must be managed by statefull processing
Control message (OpenFlow protocol)

Figure 3.1: Overview of network functions.

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.

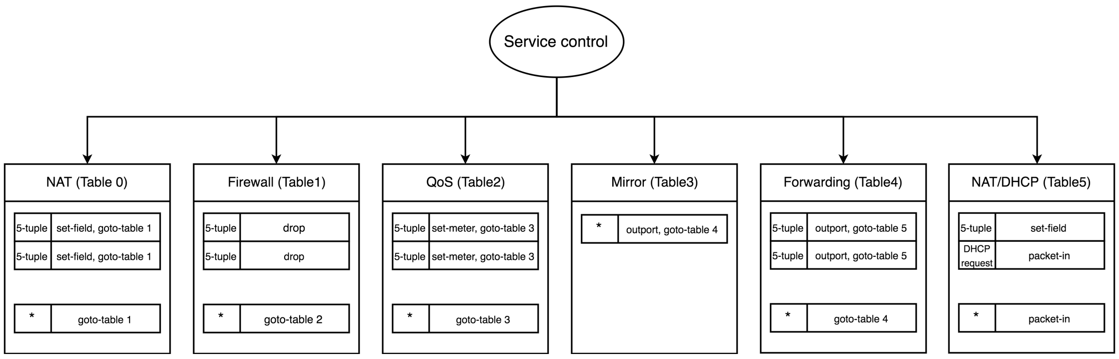


Figure 3.2: Flow table order of vCPE service.

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 the actions in matching rules. The flow table [33] 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 [27], 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.

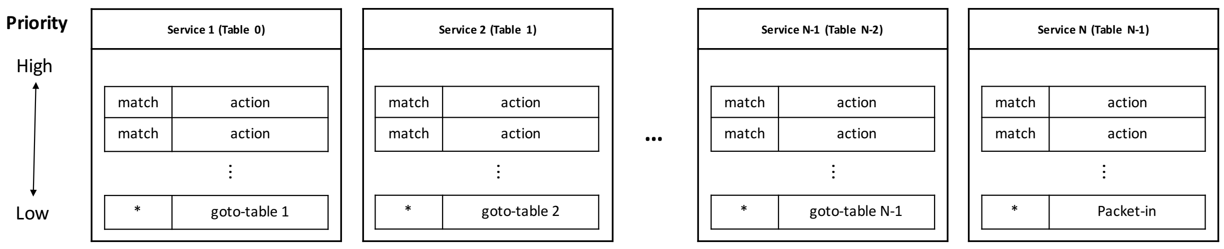
Before we discuss about the multiple flow table strategy, we introduce the pipeline of OpenFlow flow table first [6]. The processing of each packet always starts at the first flow table. When being processed by a flow table, the packet is matched the flow entries in the flow table and adds corresponding 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. In our multiple flow table management mechanism, 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.

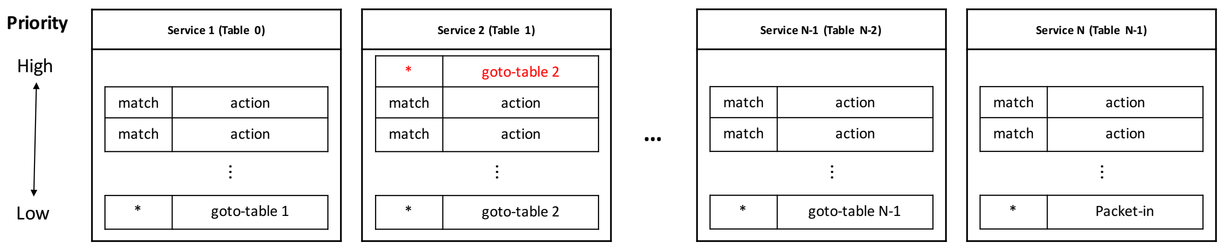
The network functions of vCPE services are the firewall, NAT, DHCP, forwarding, traffic mirroring and QoS. The order of each function was determined as shown in Fig. 3.2 (note that the flow tables are counted from zero). In the following sections, we introduce the method of implementing these network 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

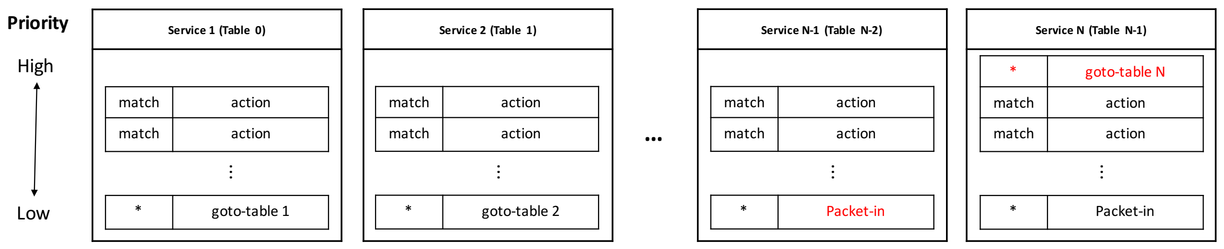
Service control is used to enable or disable services. As shown in Fig. 3.3a, a packet-in rule is always placed in the flow table of the last active service as a table miss in case there is no corresponding rule. To enable the service chain, the rules of each



(a) All Services are enabled.



(b) When service 2 is disabled, the force-ignoring rule is added into the table 1.



(c) When last service (service N) is disabled, the force-ignoring rule is added into the table N-1 and the packet-in rules is added in to table N-2.

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, a force-ignoring rule is added into the table of the service. (Fig. 3.3b) The force-ignoring rule has maximum priority with the action, “go to next flow table”. To enable a service, we just remove the force-ignoring rule.

If the service we want to disabled is located at last teble, we must not only modify the force-ignoring rule but also modify “packet-in” rules. The change is shown in Fig. 3.3c.

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 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 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.

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).

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.

3.4.2 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 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. 3.4, 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 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, 140.114.71.178, 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.

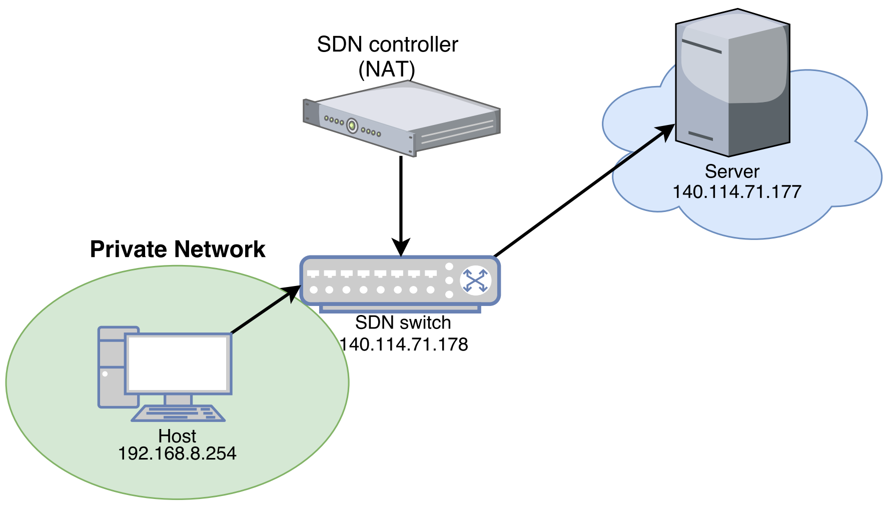


Figure 3.4: Example of modification of IP address and port number by NAT service.

Table 3.2: Flow entry for modifying the packet header fields

Direction	IP src	IP dst	src port	dst port	action
Outgoing	192.168.8.254	140.114.71.177	7878	9898	IP src $\rightarrow$ 140.114.71.178; L4 src port $\rightarrow$ 2000
Ingoing	140.114.71.177	140.114.71.178	9898	2000	IP dst $\rightarrow$ 192.168.8.254; L4 dst port $\rightarrow$ 7878

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. 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.

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.

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.

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 architecture, 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 functions are provided for QoS. We first introduce three strategies and then discuss the flow table order of QoS in the multiple table model.

3.4.6.1 Rate Limitation of Hosts

When some hosts utilize a high network bandwidth, the speed for other hosts slows or traffic congestion occurs. To prevent these effects, rate limiting is used to control the rate of traffic from a host. For implementing the host rate limitation, we first create a meter for the desired bandwidth and then add a flow, of which the match is the host's MAC address and the action is the meter.

This method is illustrated in 3.5. In T0, host 1 and host 2 are not limited yet. Because host 2 utilizes a substantial amount of bandwidth from the network in T0–T1 the network administrator sets the host 2 rate limit to 400 Kbps. When the controller receives this request, marked as (a), it creates a meter with meter id = 1 and bandwidth = 400 Kbps and sets the rule in the flow table with the destination MAC address = MAC address of host 1 and meter = 1. According to our new flow table, it limits the rate of the target host. Then, the traffic from host 2 is reduced and immediately limited to 400 Kbps. A similar

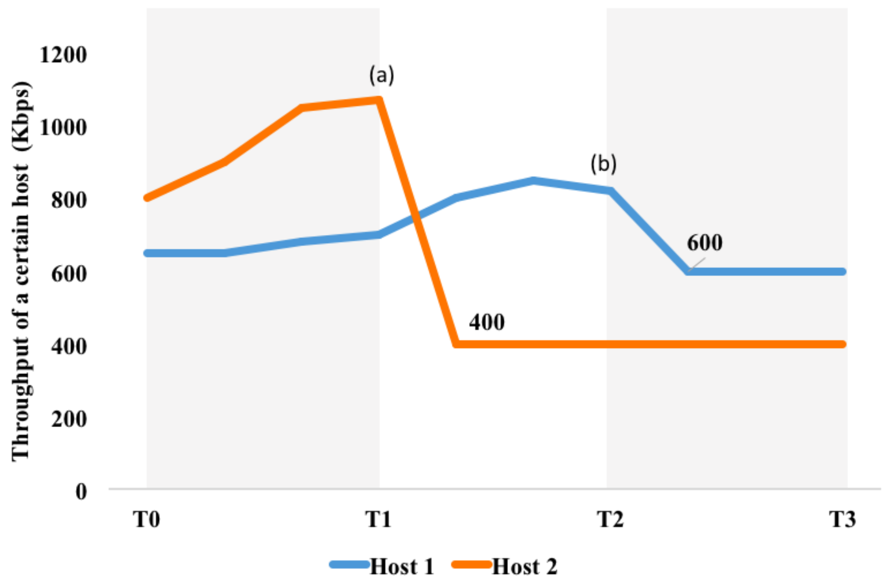


Figure 3.5: Illustration of rate limiting for a certain host

situation occurs for host 1 in T2. The administrator chooses to set host 1 under 600 Kbps, marked as (b), and then the traffic from host 1 is limited to 600 Kbps.

3.4.6.2 Rate Limitation of Applications

An increasing number of network applications such as online games, video streaming, and conference calls are used. Therefore, substantial traffic exists in the network. Consequently, we integrated a flow classification engine to identify which application the flow belongs to. The integration scenario is presented in Fig. 3.6.

To rate the limit for a certain application, we must first gather per-flow statistical information. When a connection is created, the first packet of the connection is handled by the forwarding service and the 5-tuple rules are added into the SDN switch to obtain the bandwidth information of the per-flow connection (see Section 3.4.4 for details). With the classified result of application identification, the application type of each connection can be determined.

When the network administrator requests to rate limit a certain application to a certain bandwidth, the bandwidth is equally distributed to each connection of the application. The controller sets all flows that belong to the same application into the same meter, and the bandwidth of the meter is modified to achieve the desired value. For example, suppose that we can determine which flows belong to an application through the flow classification engine. This application is to be limited to 1000 Kbps. In T0 (Fig. 3.7), three connections belong to this application, and the bandwidth of the meter to each link is set as 333 (1000/3) Kbps. In T2, two connections are added to this application, and the bandwidth of the meter to each link is reset as 200 (1000/5) Kbps. In T2, one connection is obtained in this application, and hence, the bandwidth of the meter to each link is reset as 250 (1000/4) Kbps. However, the sum of bandwidth from this application should always be 1000 Kbps. In other words, the bandwidth is dynamically adjusted.

It is worth noting that we did not change any rules in flow table; we merely change the bandwidth of the correspond meter in the meter table to reduce the overhead of switch and controller.

3.4.6.3 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. 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.

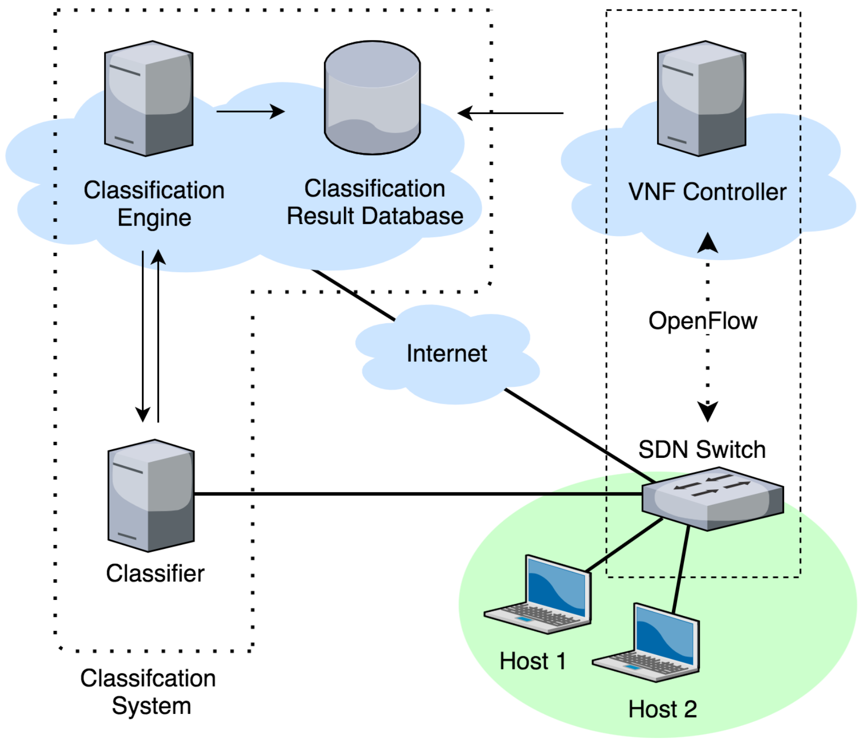


Figure 3.6: Architecture of our application identification system in classification phase.

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.

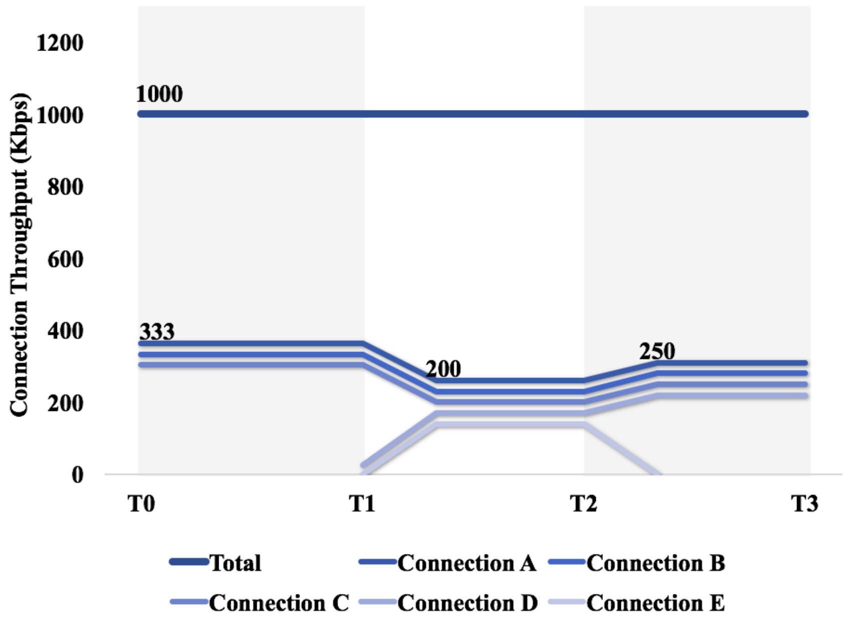


Figure 3.7: Illustration of rate limiting for a certain application.

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 anf 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 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

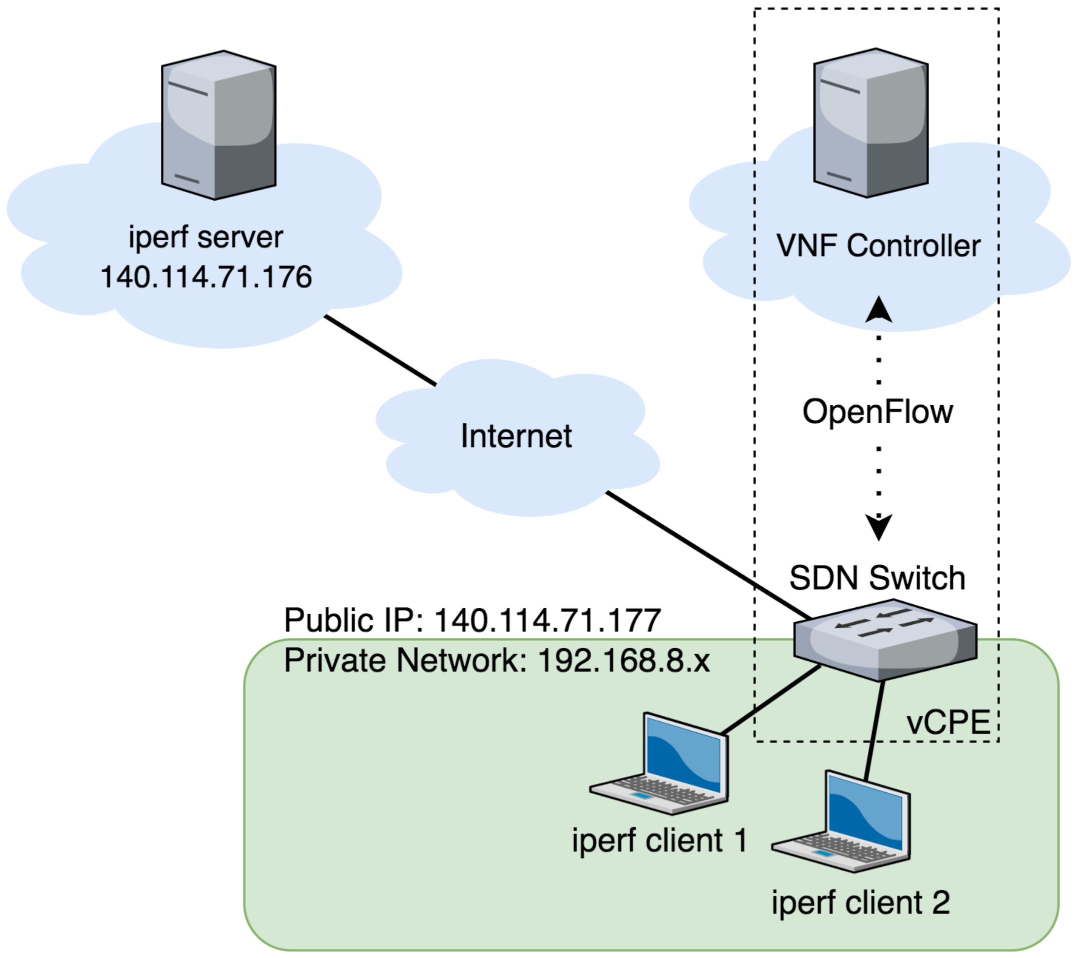


Figure 4.1: Multiple Table Performance Evaluation Scenario.

controller was run on the Dell PowerEdge R630 rack server, and the EdgeCore AS5712-54X [34] was used as the SDN switch. We used iPerf [35] 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

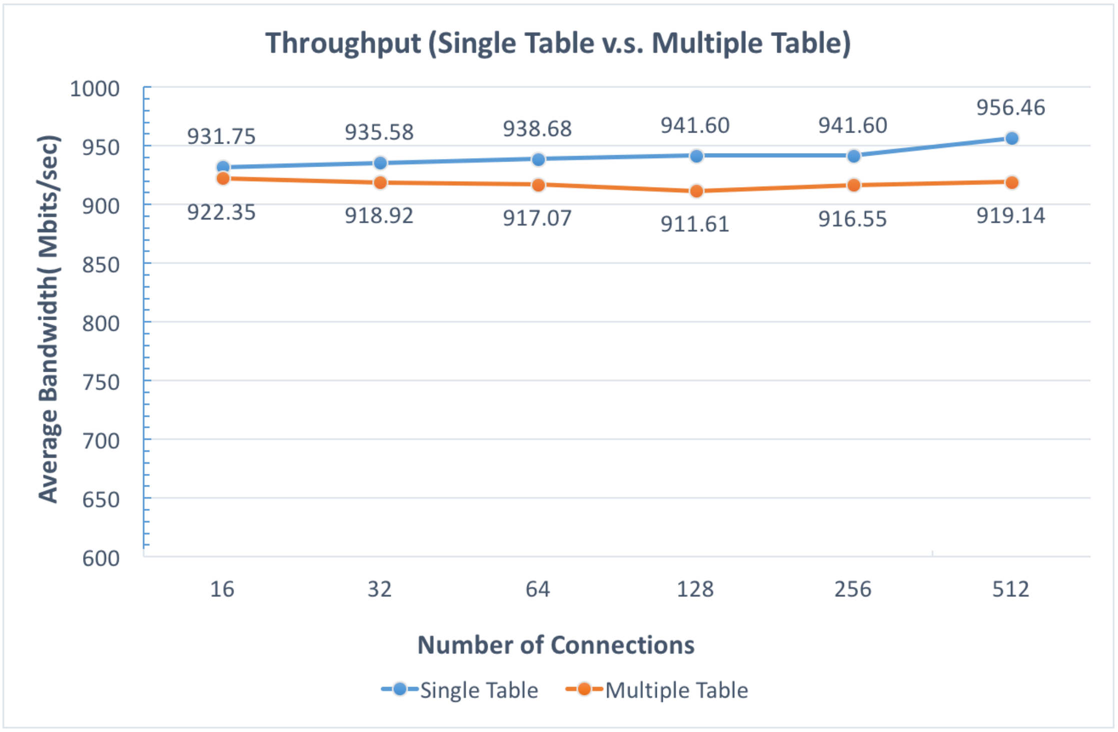


Figure 4.2: Results of Multiple Table Performance Evaluation.

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 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.

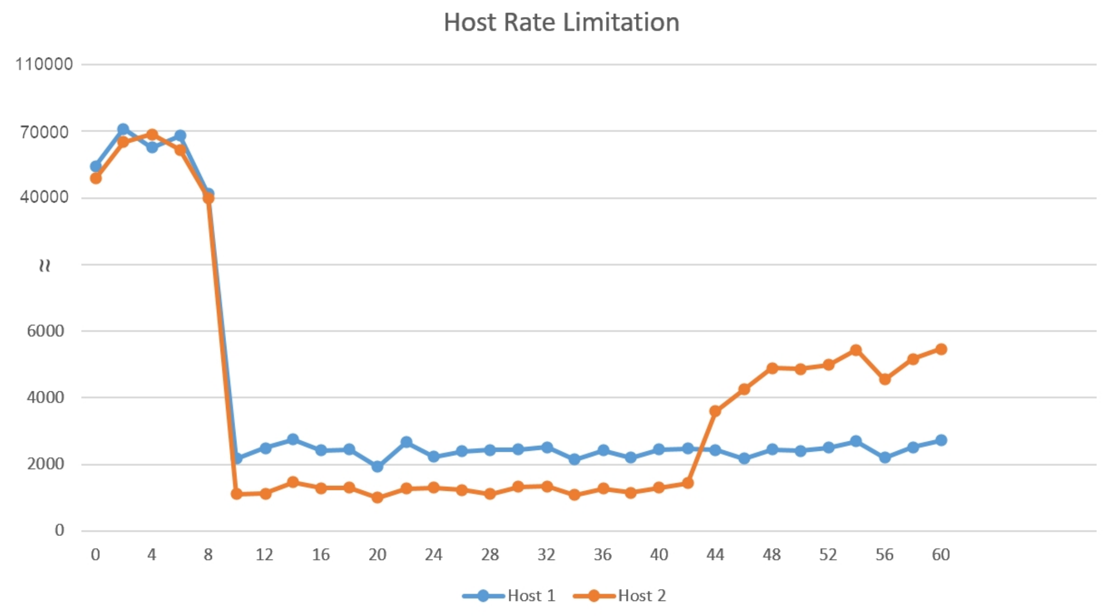


Figure 4.3: Limit the rate of a certain host.

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 [34] 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 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

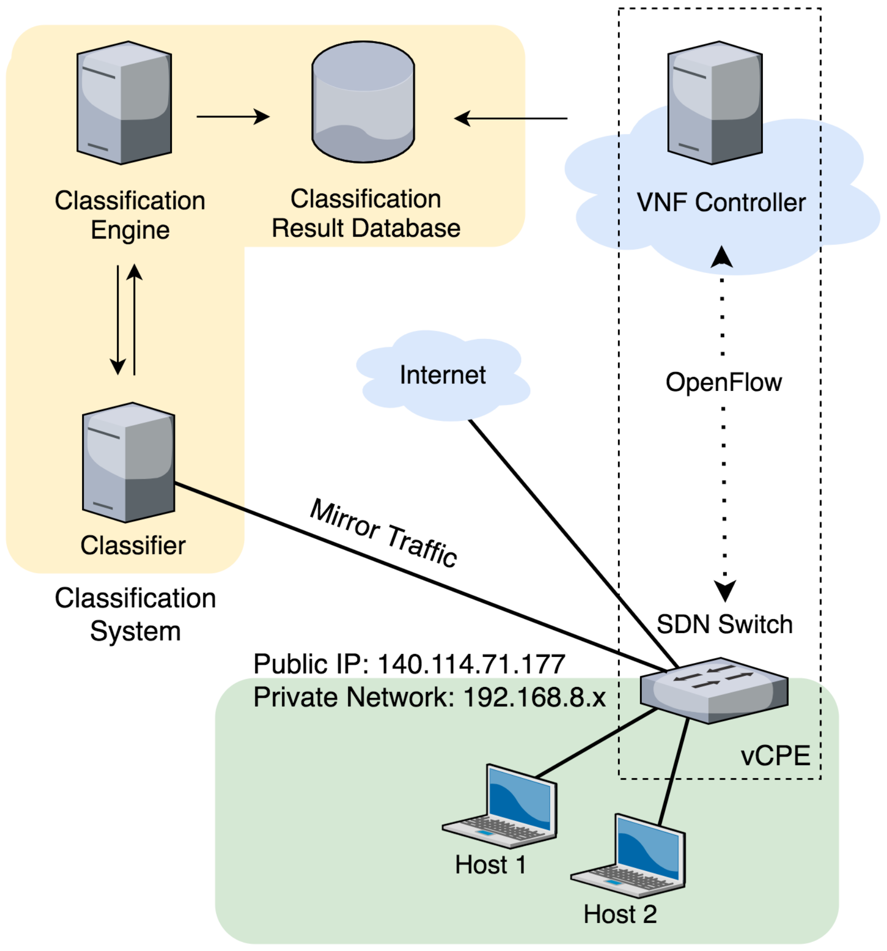


Figure 4.4: Scenario of integration evaluation.

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$^{\text{nd}}$ 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

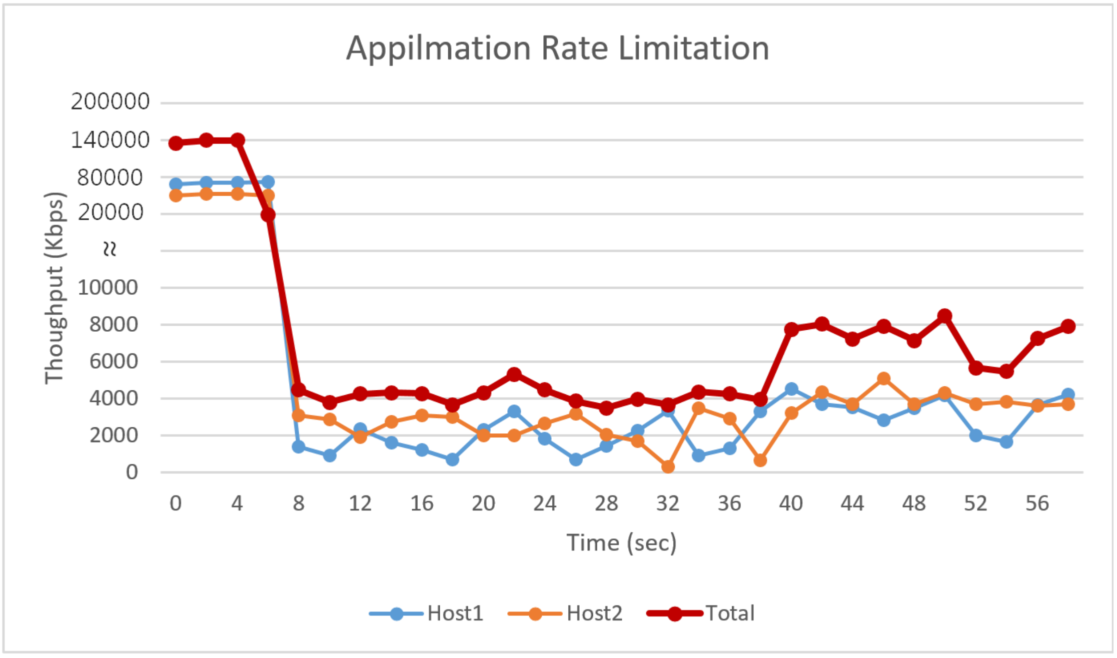


Figure 4.5: Limit the rate of Filezilla.

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 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.

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 use 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] 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, no. 2, 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, no. 1, 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, no. 2, 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*, 2012, pp. 22–24.

[8] NFV ISG, "Network Functions Virtualisation (NFV); Virtual Network Functions Architecture," ETSI, Tech. Rep. GS NFV-SWA 001 V1.1.1, 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." [Online]. Available: http://www.nec.com/en/global/solutions/tcs/vcpe/

[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," ETSI, Tech. Rep. GS NFV 001 V1.1.1, 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, Jun. 2007.

[15] ——, “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, Sep. 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, no. 2, 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] 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.

[24] 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.

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

[26] ——, “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.

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

[28] 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.

[29] 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

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

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

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

[33] 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.

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

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