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

Abstract—**This Article is about a system how to optimize a service chain in SMB Network. There are many vCPE researches on Internet but a few to discuss about organization of Openflow switch and NFVs. This paper will propose a framework to optimize and manage NFVs with multi tables of Openflow switch. Most of paper talking about how to build architecture in Core Network of ISP, this paper will focus on how to use tables in SDN switches to maximum efficiency on Internet.**

Index Terms—**Software defined networking, vCPE**

I. Introduction

THIS document is a template for Microsoft Word versions 6.0 or later. If you are reading a paper or PDF version of this doconference. By Eric

II. Related Work

By Eric

III. System Description

A. Overview of Network Function

With the concept of SDN-enabled VNFs, SDN technology work not only for traffic steering but also as a part of network functions. In this idea, network functions have been achieved by the synergies between computer and network infrastructures, shown in Fig. 1. The former is a VNF controller, mainly responsible for dealing with stateful processing. The latter is a SDN switch, used for stateless processing.

1) Stateful Processing component (VNF controller in container): This component have to control the workflow, keep the state associated with the VNF, and provide interface for service providers or customers to configure and update the behavior of the stateless datapath processing component. We use SDN controller to implement the NFV controller and its worth noting that we use southbound APIs of SDN controller framework to handle the interface between the stateful and stateless component with OpenFlow protocol, which was originally designed for this.

2) Stateless Processing component (SDN datapath): Stateless processing component, are implemented by SDN datapath resources, which is optimized for data plane traffic processing. Since SDN switch have decoupled the control plane and data plane, so it can accept the control message from the stateful processing component.

Using the advantages of this architecture, we can assign stateless or light-weight state work to the SDN switch, for example, packet filtering and packet counting, to load-off the computing resources. If we want to update our service, we just need to update the stateful component, since the stateless component just follow the command from stateful components

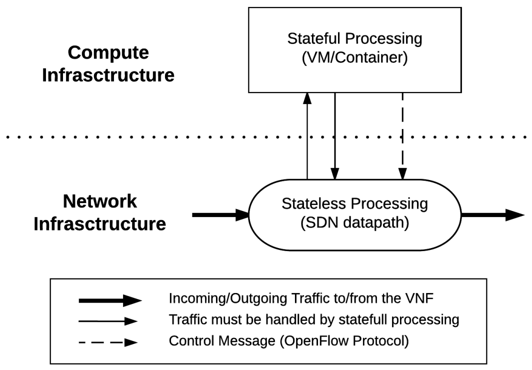
Compute Infrasctructure
Stateful Processing (VM/Container)
Network Infrasctructure
Stateless Processing (SDN datapath)
Incoming/Outgoing Traffic to/from the VNF
Traffic must be handled by statefull processing
Control Message (OpenFlow Protocol)

Fig. 1: Overview of network function.

B. Service Deployment Model

With architecture mentioned in Fig. 1, we come up with a network function service deployment model. Because computing infrastructures handles the algorithm and policies, and the generic network devices only do stateless processing, the customer just need to buy a general SDN switch at their home gateway and will have a different network function service by subscribing different the NFV controller through our vCPE platform.

Fig. 2 illustrates the service deployment model. Each green area is a local network domain of customer. At the gateway of this domain, theres a SDN-enabled switch. The customer can subscribe to our vCPE service through our dashboard. After subscribing, the vCPE system will create a new docker container, in which running a SDN controller we developed. The customer only need to setup the gateway SDN switch to connect the SDN controller by the OpenFlow protocol, then the switch will handle these service.

C. Architecture of the vCPE system

The architecture as shown in Fig 3 including of a Infrastructure Controller, Infrastructure Orchestrator, Cloud Database, VNF Controllers and VNF Orchestrator. The Infrastructure Orchestrator, VNF Orchestrator and Cloud Database are web servers Each component is introduced in the subsection below.

1) Infrastructure Controller: The infrastructure controller is a composed Docker management server with the ability to manage the Docker resources like containers and images. The infrastructure doesn't handle the customer authentication or maintaining the state of running service, it just follows the

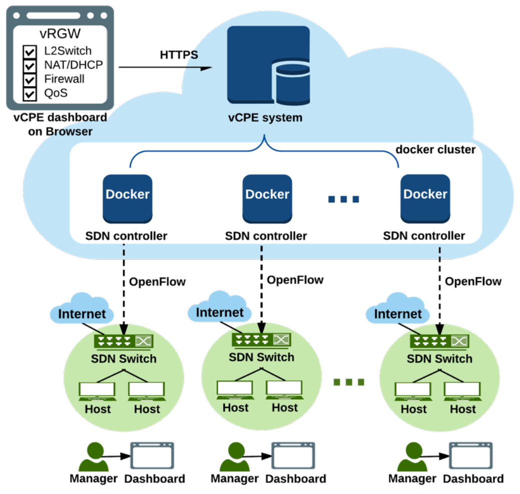

Fig. 2: Service deployment model.

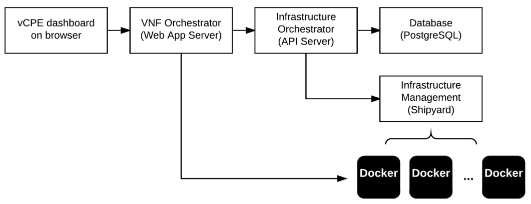
vCPE dashboard on browser
VNF Orchestrator (Web App Server)
Infrastructure Orchestrator (API Server)
Database (PostgreSQL)
Infrastructure Management (Shipyard)
Docker
Docker
...
Docker

Fig. 3: Architecture of the vCPE framework.

request from the infrastructure orchestrator to create, delete, start, stop and inspect containers.

2) Infrastructure Orchestrator: The infrastructure orchestrator plays the key role of our system. It connecting and automating of workflows when we deploy our services. When a customer subscribes, the infrastructure orchestrator authenticates the customer first, next it will call the infrastructure controller to create a container for this customer, and update information in database afterwards. It handle the entire lifecycle of our vCPE service.

3) Cloud Database: The cloud database is used for restoring the of our vCPE services, which include each customers credential, customers container settings and virtual CPE service states. The cloud database is using PostgreSQL, which is a open source, easily customized and object-relational database system. Only Infrastructure Orchestrator has permissions to access cloud database.

4) VNF Controllers: VNF Controllers contains a SDN controller developed with ryu framework and a remote launcher module. The SDN controller does not have a remote launcher module to remotely execute a SDN controller. We built a lightweight server as a launcher module to resolve the remotely execution issue. The remote launcher module monitor the SDN controller process ID (PID) and properly kill the SDN controller process ID when on demand. When the infrastructure controller once create the container, the remote module will run up initially, waiting request from VNF Orchestrator. The details of SDN controller design will be presented at section IV.

5) VNF Orchestrator: The VNF Orchestrator is a Web application server hosting on Amazon Web server, being online for customer and provide a dashboard for virtual CPE and containers management and configuration. Through the web UI provided by the VNF Orchestrator, the customers can subscribe to the desired service and without typing any command via the command line interface (CLI). After receiving the subscribing message, the VNF orchestrator will request the infrastructure orchestrator to create a new VNF controller, and then send the virtual CPE configuration to the new VNF controller. Based on configuration demands under different conditions, the network administrator is able to select any of the listed network functions on the dashboard such as Firewall, NAT, DHCP and QoS management.

D. Network Functions

The proposed vCPE service provides following network functions.

1) Firewall: The firewall service could filter the packets based on packet header fields, including MAC address, IP, and layer4 protocols. The network manager can add new rules or remove rules to the access control list through our vCPE GUI.

2) DHCP: DHCP service allocates IP addresses to hosts dynamically. To implement the function, the SDN controller cope with UDP packets which use port 67 and 68. It generates DHCP offer and DHCP acknowledge to reply DHCP discovery and DHCP request received from the hosts.

3) NAT: The NAT service is a network function that can remap IP address to another one. The Source NAT (SNAT) is typically used by internal users (inside private network) to access the Internet (outside network). The network function uses the action Set-Field, which defined by OpenFlow protocol for rewriting packet header fields. Via vCPE GUI interface, the network manager can set up the WAN port of SDN switch, public IP, default gateway, and local network address when using this function.

4) QoS: Quality of Service (QoS) are always used to control the traffic flows of a network and prevent the traffic to exceed the network capacity and cause traffic congestion. Therefore we implement the bandwidth management using meter which is defined within OpenFlow protocol 1.3 to set the limitation of the bandwidth. Beside achieving network functions virtualization with SDN technology, we also make the network administrator manage and monitor a network more easily. As a result, we can offer the user the best network quality in the limited network resource without traffic congestion. In this paper, our QoS integrates with a flow classification engine and offers the three ways of bandwidth management.

- For a specific host.
- For a specific application.
- For a specific application from a host or a domain.

5) Forwarding: Forwarding Service is a basic service that forward traffic to its destination and we use the Mac address learning concept to implement our forwarding function.

E. Application Identifier

Application identification is to identify each flow as an application name to enable QoS management system to do bandwidth control or distribution at application level. A flow is defined by 5-tuple (source IP, source port, destination IP, destination port, and transport layer protocol). Applications include desktop applications, native mobile applications, and web applications, such as Facebook, Skype, YouTube, Instagram, Line and WeChat. We use supervised machine learning and a method based on inspecting domain name service (DNS) responses to do flow classification. After application identification system classifies a flow as an application name, it sends the classification result to a server with database. The server stores the classification result into database, and waits for query with 5-tuple from QoS management system. The detail will be described in section V.

IV. Network Function With Multiple Flow Table Management Model

A. Multiple Flow Tables Strategy

B. Service Control

C. Firewall

D. DHCP

E. NAT

F. QoS

G. Forwarding

V. Application Identification Method

In order to let QoS management system be able to control or distribute bandwidth at application level, we designed and implemented an application identification system (also called flow classification system) to judge each flow is established by what application. In other words, the system can classify each flow as an application name in real time, rather than a rough category or a transport layer protocol. We use supervised machine learning (ML) and a method based on inspecting domain name service (DNS) responses to do flow classification. Except for DNS responses, the system only analyzes transport layer information of packets without inspecting their payload.

A. Architecture of application identification system

There are training phase (also called preparation phase) and classifying phase, their architectures are shown in Fig 4 and Fig 5 respectively. These architectures can be integrated, and do training and classifying in the same time.

B. Overall procedure of application identification

In training phase, we generate some data needed by ML and the method based on inspecting DNS responses. In classifying phase, if a flow meet some requirement, we use the method based on inspecting DNS response to classify; otherwise, we use ML to classify.

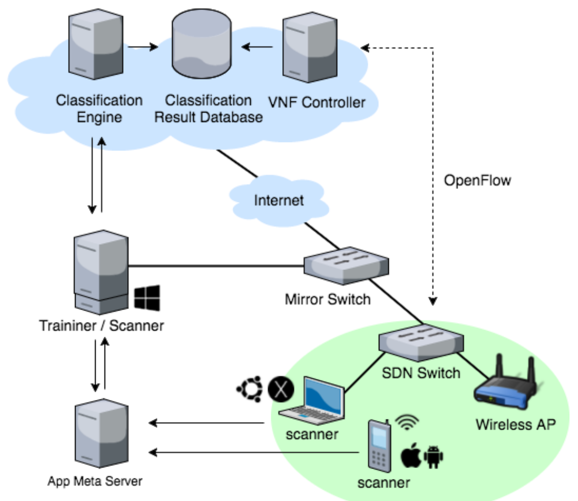

Fig. 4: Architecture of our application identification system in training phase.

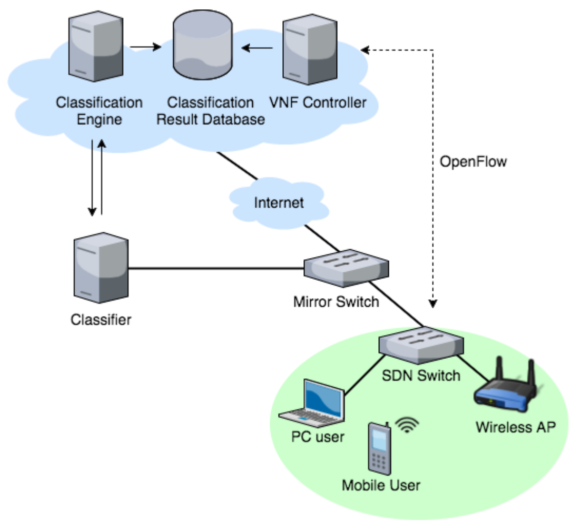

Fig. 5: Architecture of our application identification system in classifying phase.

C. Procedure of Supervised ML in our system

In training phase, the architecture is show in Fig 4. Flow classification engine is to build classification model by training data from Trainer, and update the model when new training data comes. Trainer is to analyze traffic from mirror port to get flow attributes. After Trainer finishes calculating flow attributes of a flow, it gets ground truth of the flow from Scanner or App Meta Server to generate training data, and sends training data to flow classification engine. App Meta Server is to store mappings of 5-tuple and ground truth from Scanner, and accept queries from Trainer. Scanner is to get mappings of 5-tuple and ground truth from OS. In classifying phase, the architecture is shown in Fig 5. Flow classification engine is to use flow attributes from Classifier and classification model to get a classification result, and sends classification result to Classifier. Classifier is to analyze traffic from mirror port to get flow attributes. After Classifier finishes calculating flow attributes of a flow, it use flow attributes to ask flow classification engine for classification result if needed.

D. Flow attributes we adopted in ML

In ML, when system calculates flow attributes, we only use transport layer information of packets which have payload. We use Application Round (APPR) proposed by [] to analyze flow behavior. We use number of packets, packet sizes, transmission time, transmission direction, throughput, and APPR to define flow attributes. There are 69 attributes in total, as defined in our previous work

E. Algorithms we adopted in ML

We use algorithms implemented by Weka, a famous open source ML tool developed at the University of Waikato. We call Weka Java API to build classification model. We use three algorithms, namely RandomSupSpace with RandomTree [], FilterClassifier with discretization and RandomTree [], and RandomCommittee with RandomTree [].

F. Procedure of a method based on inspecting DNS response in our system

In training phase, we collect the mappings of server IP and application name. A mapping of server IP and application name means the application has ever establish connection with the server whose IP is the server IP, and the server IP has ever occurred in any DNS response we have captured. In classifying phase, we only use those mappings which have been used by one application to classify. When system detect a flow, we check whether source IP or destination IP of the flow match any server IP of those mappings. If it matches any server IP, we regard corresponding application name as classification result; otherwise, we use flow attributes of the flow and classification model to classify.

VI. PERFORMANCE EVALUATION

VII. CONCLUSION AND FUTURE WORKS