# NFV blah

Chi-Hsuan Li[†], Che-Wei Lin[‡], Sheng-Jung Wu[†]

[†] Department of Computer Science, National Tsing Hua University
No.101, Sec. 2, Guangfu Rd., East Dist., Hsinchu City 300, Taiwan
[‡] Institute of Communication Engineering, National Tsing Hua University
No.101, Sec. 2, Guangfu Rd., East Dist., Hsinchu City 300, Taiwan

**Abstract** IEICE (The Institute of Electronics, Information and Communication Engineers) provides a template file for the IEICE Technical Report.

**Key words** IEICE, IEICE technical report, LaTeX, template

## 1 Introduction

In these days, the network functions have took its place in modern life, play an important role in providing critical performance, security and policy compliance capabilities. However, network service is usually achieved by the hardware and using a dedicated device to deploy and integrate a variety of network functions, which causes the lack of flexibility in service delivery. When the network service provider wants to introduce the new service, customers will tend not to replace with new equipment and the service provider must also cost high capital expenses (CapEx) and operation expenses (OpEx) to replace the physical equipment. As a result, the network service is difficult to promote and update.

It was against this background, the concept of network function virtualization (NFV) was put forward, expecting to solve these problem. NFV transforms how network operators architect their infrastructures with the virtualization technology to separate software instance from hardware platform, and implements virtualized network functions (VNFs) through software techniques and run VNFs on compute infrastructures. NFV also innovates in the service delivery, by deploying these functions over general virtualized compute infrastructures, which are used to replace the traditional network appliances. When these functions are no longer a large degree depent on specific physical devices, the management of VNFs can be easier and the deployment will achieve flexibility and scalability.

With the trend of Software Defined Network, virtualization of the network functions is a big step forward. SDN works perfectly with virtual network functions. From the very beginning SDN simply provides dynamic links between network functions. Which represents the service provider or cloud center managers can dynamically set the target traffic need to go through what kind of processing and processing order. So that traffic can be separated through data plane and control plane by SDN technology and resiliently steer between services. And to the recent SDN technology can not only provide connectivity, but also can become an important part of the realization of virtual network functions. In this paper, these network functions is called SDN-enabled VNF.

The SDN-enabled VNF

This paper proposed a number of implementation of the SDN-enabled network functions and build up a NFV with the management system. To explore the feasibility of this SDN to participate in the network functions, also to provide network service providers a complete solution concept.

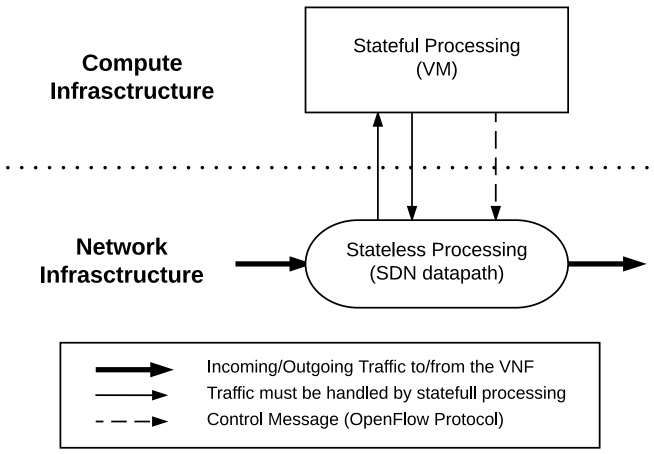
Compute Infrasctructure
Stateful Processing (VM)
Network Infrasctructure
Stateless Processing (SDN datapath)
Incoming/Outgoing Traffic to/from the VNF
Traffic must be handled by statefull processing
Control Message (OpenFlow Protocol)

Figure 1: NFV design overview

## 2 Related Work

## 3 System Design and Implementation

### 3.1 NFV Design Overview

With the concept of SDN-enabled VNFs, the network functions have been achived by the synergies between compute and network infrastructures. The former is mainly responsible for dealing with stateful processing, and the latter is used for stateless processing component.

#### 3.1.1 Stateful Processing Componemt

This component have to perform more complex algorithm, 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, since software is good at these tasks. It's worth noting that we use southbound APIs of SDN controller to handle the interface between the stateful and stateless component with OpenFlow protocol, which was originally designed for this.

#### 3.1.2 Stateless Processing Componemt

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

## 3.2 Architecture of the vCPE Platform

### 3.2.1 System Overview

### 3.2.2 System Implementation

# 4 Virtualized Network Functions and Use Cases

In this section, we would introduce the SDN-enabled VNFs we have implemented. Considering this approach, the NFVs can be furthur classified into three categories with different state complexity when processing incomming traffic:

1) **Policy-based service:** In this category, the network infrastructure takes the primary responsibility of handling incoming traffic and the traffic need only stateless processing which is following certain policy given by the service management.

2) **Dynamic-configured service:** The first a few packets are needed to send to stateful processing component for initializing configuration, and the rest packets can be handled by SDN datapath with the previous configured setting. For example, NAT and DHCP services are belongs to this category.

3) **QoS service:** In this approach, our NFV is collaborate with a traffic classification system. The whole incoming traffic will be redirect to the compute infrastructure, where the classifiers run on. When we get the information abount current connection and traffic, we can take responding action, like dropping the connection of malware or limited the throuput of certain streaming application.

## 4.1 Policy-based service

### 4.1.1 Firewall

Our virtual firewall service provides the usual packet filtering. It filters each packet based on information contained in the packet itself, using a combination of the packet's source address, destination address and its protocol, which can be transformed into a list of OpenFlow rules.

In this case, the target traffic need only stateless processing which can be done solely by the SDN datapath resource. Therefore, we just provide policy management interface to handle the request from service manager in the software controller component. When the firewall network service was requested to update firewall policy, the controller will map the request to OpenFlow protocol messages and send to the SDN switch, which will follow the policy to process the incoming traffic.

For example, if the service manager choose to block certain protocol for source IP address 192.168.0.100, he can send the request to controller via management API. The request will be mapped to the form of OpenFlow protocol, Match-Fields with the drop action, and then

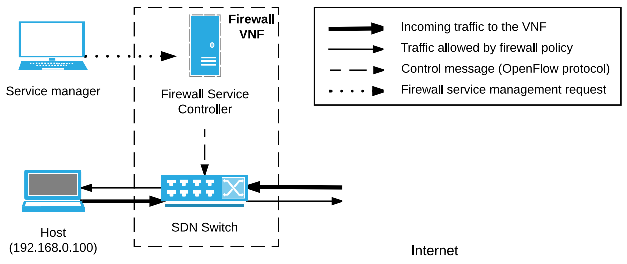
Firewall VNF
Service manager
Firewall Service Controller
Host (192.168.0.100)
SDN Switch
Internet
Incoming traffic to the VNF
Traffic allowed by firewall policy
Control message (OpenFlow protocol)
Firewall service management request

Figure 2: Firewall use case

send to the datapath resource. As a result, when the host 192.168.0.100 want to send SSH request, it will be blocked by the SDN switch.

### 4.1.2 Network Tap

## 4.2 Dynamic-configured service

### 4.2.1 NAT

The NAT (Network Address Translation) service is a virtualized network function which remapping one IP address space into another by modifying network address. The The remapping process is with Ryu SDN controller. The OpenFlow protocol optional action Set-Field is employed to rewrite packet header fields to easily convert the forwarding policy into OpenFlow entries. Currently, the Source NAT (SNAT) is implemented which allows multiple hosts on the private network (inside) to connect to the Internet (outside).

When a packet from private network host arrives at the SDN switch, the packet header fields are modified by Set-Field action. For outgoing packet, the source IP address and source port number are modified to fit the public IP of NAT and remap a port number of NAT. For ingoing packet, the destination IP and destination port number are modified accordingly. Below are some examples to present the flow tables for NAT service with ingoing packets and outgoing packets.

As demonstrated in Table **??**, the public IP address of NAT is 10.10.10.1 and a host (client) with IP address 192.168.8.40 and port number 5566 sent a packet to a server with IP address 8.8.8.8 and port number 7788. The action modifies the source port to 2000 which means the NAT is enabled by the SDN controller. As shown in Table **??**, when the server sends back several packets to the client, the destination IP and destination port number are modified by the SDN switch accordingly.

### 4.2.2 DHCP

## 4.3 QoS

### 4.3.1 IDS Integration

z

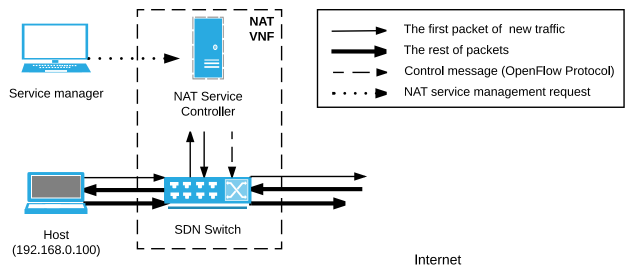

Figure 3: NAT VNF use case

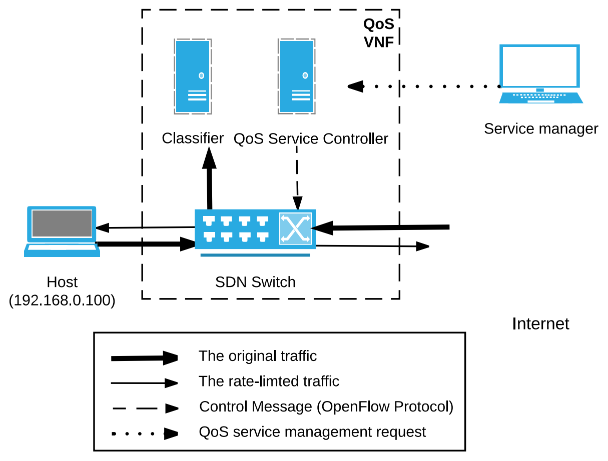

Figure 4: QoS VNF use case