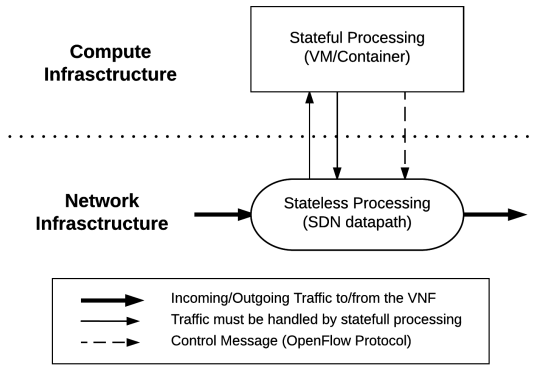
Fig. 1. Overview of network function



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. vCPE System Overview

A. Overview of Network Function

With the concept of SDN-enabled[1] VNFs, SDN technology work not only a for traffic steering but also as a part of network functions. In this idea, network function have been achieved by the synergies between compute and network infrastructures. 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 statful component, since the stateless component just follow the command from stateful components.

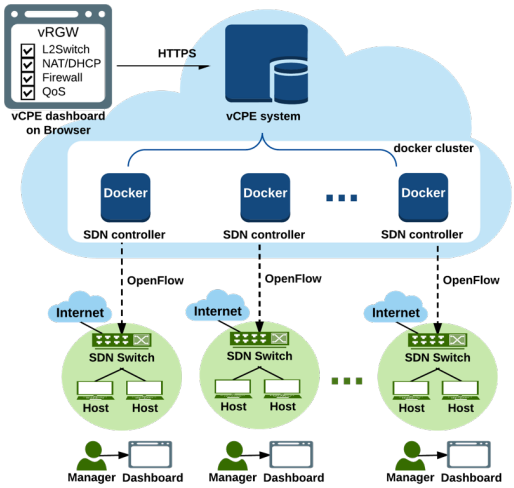


Fig. 2. Service Deployment Model

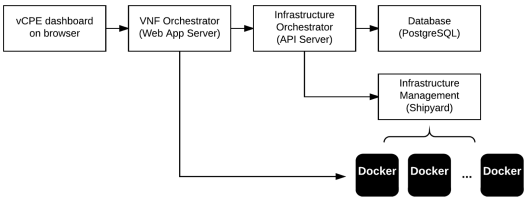


Fig. 3. Overview of vCPE framework

B. Service Deployment Model

With architecture mentioned in III-A, 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 NFV controller through our vCPE platform.

Figure 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 framework

1) System Overview: 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.

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

3) 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.

4) 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.

5) 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.

6) 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

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) 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.

3) 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.

4) 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.

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

E. Application Identification

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 table management model

A. Multiple Flow Tables Strategy

In subsection III-A, the vCPE service design architecture have been introduced. The network functions are handled by the cooperation between SDN controller on cloud and SDN switch at the local network gateway. The controller transform the network functions to series of OpenFlow rules requests and send to SDN switch. Following the orders from controller, the

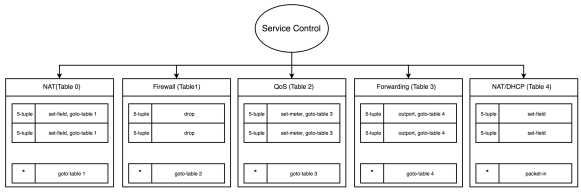


Fig. 4. The flow table order of our vCPE service

SDN switch inserts rules to its flow tables, checks incoming packets against the flow entry match fields, and execute the actions in matching rules. The flow table defines all matching and corresponding processing, thus playing an important role to executive network function.

Since the flow table is crucial component, and we find that single table binds us to implement our network functions. The [2] also mentioned two condition for single table is too restrictive. The first is a single packet need to perform independent actions based on matching different fields. The second is that the packet needs two-stage processing. Involves into both situation, our network functions implemented by multiple flow tables strategy.

In multiple flow tables strategy, the most important question is: which flow table should we insert rules into? We use the network function as a demarcation, that is, SDN applications which are responsible for specific network functions will only insert rules to one specific flow table, so we can focus on the design of the network function itself. In this way, however, the order of flow table become crucial. Should we put this network function at first, or the other? The answer is about the type of match and action in the rules generated by the network function

The network functions of our vCPE services including Forwarding, Firewall, NAT, DHCP and QoS. We have determined the order of each function, shown in 4. In the following subsections, we will introduce how to implement these network functions, which type of rules will be inserted to SDN switch, and how these rules affect our decision of the order of flow tables.

B. Service Control

Service control is used to enable or disable service. To enable a service, we need to modify the table-miss rule. We always put a packet-in rule in the table of the last active service as a table-miss in case that there isn't any corresponding rule. To make our service chain possible, the rules of each service except the last service contains an additional action, go to next table, so the packets can continue to pass through all active services.

To disable a service, we not only need to modify the table-miss rule but also have to add an enforce rule. Each enforce rules has maximum priority and the action is goto next table. It means that packets will still pass through the disabled service's table, and the only thing they do in this table is ignoring other rules and go to next table.

Fig. 5. Illustration of NAT service

C. Firewall

Firewall service is able to block traffic dynamically, and in this service, the packets will not cause any packet-in event. On the dashboard, we can specify the blocking policies. There are 4 kinds of policies:

- Block any traffic from a certain source IP.
- Block all traffic to a certain destination IP.
- Block traffic based on known layer 4 protocols, such as SSH, HTTP, etc.
- Block traffic to any 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 will be installed immediately. Then any traffic that hit the blocking rules will be dropped. For normal traffic, they will not be affected.

As shown in Table I, all the actions of flow entries are drop. The 1st rule illustrates that SSH connection with source IP address 192.168.2.1 would be blocked. The 2nd rule shows the flow entry would block the Telnet protocol.

In our multiple table model, the firewall service is located in the table 1, since once packets are caught by the blocking rules, they doesn't need to apply any other services. The packets which match the rules will be dropped immediately, and their journey in the flow table ends here. To other no-blocked packets, they pass all blocking rules and finally match the table-miss rule, which will let the packets go on the next table. The action of firewall is different from other services, since in other services, no matter what actions are taken to the packets, the packets have to go to the next table

D. DHCP

E. NAT

The NAT service could allow a lot of hosts to use one public IP address to connect the network. In order to achieve this thing, SDN controller have to set the packet header filed. Because the SDN switch would set the field, the first packet have to go to controller. And controller would add the flow to SDN switch, after that the packet would not go to controller. It can decrease the burdens of the controller.

Below is a examples to show how the NAT service modify the IP address and port number, as shown in Fig 5.

For outgoing packet, SDN switch doesnt have any flow entry in flow table, so the Packet-in event will be triggered at the beginning. The packet that is sent by private network host will be sent to SDN controller, and the packet header fields are modified by Set-Field action. The source IP address and source port number of outgoing packet will be modified to public IP address and remaped a new port number for NAT. For ingoing packet, the destination IP and destination port number are modified to fit private IP and port number. And then SDN controller add these flow entries to SDN switch, it cant avoid that all the packet sent to controller. Fig. 5 show the public IP address of NAT is 140.114.71.178 and a host

TABLE I
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 (match any value).

private IP is 192.168.8.254 and port number 7878. The client sent the packet to a server with IP address 140.114.71.177 and port number 9898.

As shown in Table II, when host sent the packet to server (outgoing), the Packet-in will be trigger, and then packet to controller. The Set-Field action would modify the Source IP to public IP address of NAT is 140.114.71.178 and source port to 2000. when the server sends back to the client (ingoing), the packet header field would be modified. The destination IP address and destination port number would be modified to 192.168.8.254 and 7788.

In the single table framework, we have to add the two rules to SDN switch to match the outgoing and ingoing situation. At the first, we predict the NAT service to be put one the last table in the multiple table framework. Because the NAT service need to set the packet header fields and play a role of connection to the outside network. And the most important thing is that the SDN switch is according to order of table to match. We have to consider the outgoing and ingoing situations. As a result of all the above factors combined, the NAT service is located at the first and the last table in our multiple table framework.

F. QoS

G. Forwarding

In this service, when the first packet in a new connection incoming, it will cause a packet-in event because there is no corresponding rule. When controller receives the packet, it will record IP-layer information, including source IP, destination IP, input port number, source mac address and destination mac address. With the recorded information, the controller is able to install a 5-tuple forwarding rule for this connection and following packets dont need to packet-in again. The 5-tuple is source IP, destination IP, network layer protocol, source layer 4 port, and destination layer 4 port.

In order to gather per-session statistics information, 5-tuple rules are needed. This is why we dont add rules based on mac address. The controller installs a pair of dummy rules for every connection, and then request the switch to get current flow statistics every second. In this way, we can get the real-time bandwidth statistics of each connection by just subtracting the byte count from byte count of last second.

V. APPLICATION IDENTIFIER 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

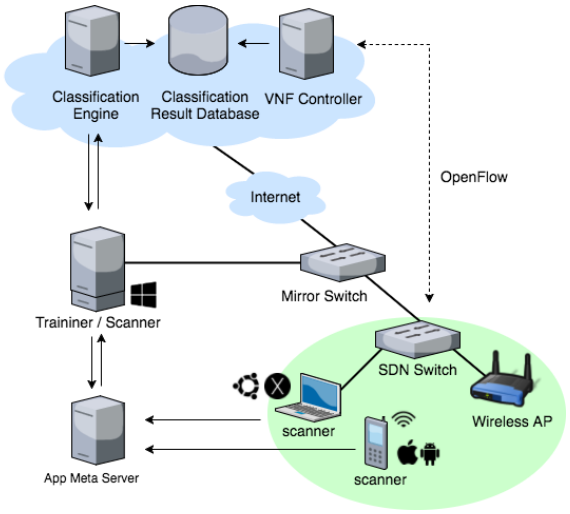


Fig. 6. Architecture of our application identification system in training phase

A. Architecture of our application identification system

There are training phase (also called preparation phase) and classifying phase, their architectures are shown in Fig 6 and Fig 7 respectively. These architectures can be integrated, and do training and classifying in the same time. The detail will be mentioned in the following paragraph

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.

C. Procedure of Supervised ML in our system

In training phase, the architecture is show in Fig I. 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

TABLE II
FLOW ENTRY FOR MODIFYING THE PACKET HEADER FIELDS OF PACKETS

	IP Src	IP Dst	L4 src port	L4 dst port	Action
1. outgoing	192.168.8.254	140.114.71.177	7878	9898	IP Src → 140.114.71.178; L4 src port → 2000
2. ingoing	140.114.71.177	140.114.71.178	9898	2000	IP Dst → 192.168.8.254; L4 dst port → 7878

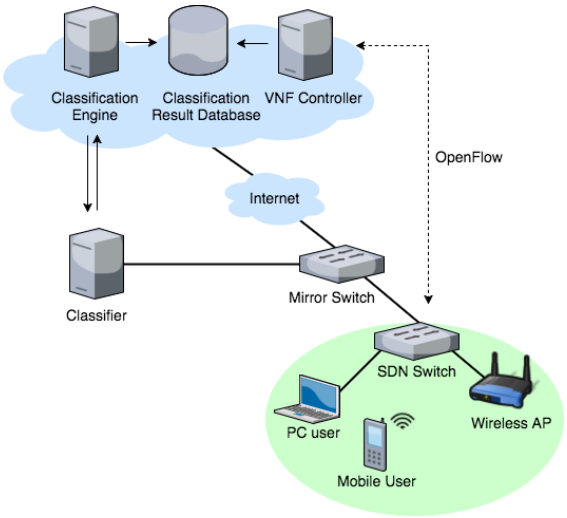


Fig. 7. Architecture of our application identification system in classifying phase

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. 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.

REFERENCES

[1] J. Matias, J. Garay, N. Toledo, J. Unzilla, and E. Jacob, "Toward an SDN-enabled NFV architecture," *IEEE Communications Magazine*, vol. 53, pp. 187–193, apr 2015.

[2] O. N. Foundation, "The benefits of multiple flow tables and ttps," tech. rep., 2015.