Service Function Chaining
Internet Draft
Intended status: Informational
Expires: January 1, 2015

C. Huang
Carleton University
Jiafeng Zhu
Huawei
Peng He
Ciena

July 1, 2014

SFC Use Cases on Recursive Service Function Chaining
draft-huang-sfc-use-case-recursive-service-00.txt

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on September 3, 2014.

Copyright Notice

Abstract

Service function chaining (SFC) provides various services that can be tailored to different requirements from diversified user groups, where each user group forms a collective client that requires similar service. SFC is typically deployed as a service overlay with its own service topology on top of existing network topology. This kind of virtualized structure naturally enables recursive service relationship where a client may become a service provider and resell SFC services to its own user groups. This document describes some exemplary use cases that show the usage of recursive (e.g. nested) service function chaining relationship.

Table of Contents

1. Introduction

New services such as service function chaining (SFC) are becoming popular with network function virtualization. Traditionally a service chain consists of a set of dedicated network service boxes such as firewall, load balancers, and application delivery controllers that are concatenated together to support a specific application. With a new service request, new devices must be installed and interconnected in certain order. This can be a very complex, time-consuming, and error-prone process, requiring careful planning of topology changes and network outages and incurring high OPEX. This situation is exacerbated when a tenant requires different service sequences for different traffic flows or when multiple tenants share the same underlying network.

Today’s SFC takes a new approach built upon network function virtualization (NFV). It involves the implementation of network functions in software that can run on a range of industry standard high volume servers, switches, and storage. Through NFV, service providers can dynamically create a virtual environment for a

specific service chain and eliminate the dedicated hardware and complex labor work for supporting a new service function chain request.

One of the great potentials NFV can enable is the capability to support recursive SFC service. A client of SFC service can resell customized SFC services to its own user groups, where the client becomes a service provider and its subscribed user groups become new clients, without adding any dedicated hardware. This kind of recursive (or nested) service relationship is quite common in daily life. Big wholesalers can sell products to smaller wholesalers and the smaller wholesalers then sell those products to other small wholesalers or directly to end users. In telecom area, the Carriers' Carriers concept is defined in RFC 4364[1], which comes from similar idea. Forming recursive business relationship has been proven to be a successful business model due to the flexibility and efficiency it provides. The same arguments can also be applied to SFC service providers.

A distinguished characteristic of recursive SFC service is that each level of service provider has its own administrative authority built over the virtual environment provided by the lower level, leading to a hierarchy of administrative levels. This kind of hierarchical structure poses both opportunities and challenges for service providers. In a later section, a use case will be presented to illustrate a specific application scenario.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119.

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

3. Use Case

There are numerous use cases that recursive SFL service can be applied to. A typical use case is described below.

Consider a scenario where Enterprise B outsources its enterprise network to a datacenter operated by a cloud service provider A. This type of scenario has been widely considered as one of the major applications of cloud computing. It is believed that enterprises can save their costs and improve their IT services by exploring the

elasticity and dynamic sharing nature of a datacenter environment. Different enterprises typically have different requirements about their outsourced enterprise networks in terms of topology and service function.

Consider the case that Enterprise B requests its outsourced network to have mesh topology with each node dedicated to a special service function. After receiving the request from Enterprise B, Cloud Provider A will create all requested virtual service function nodes with a mesh topology out of its infrastructure. Provider A will also need to assign an ID which is unique in his authority to identify this mesh service function chain.

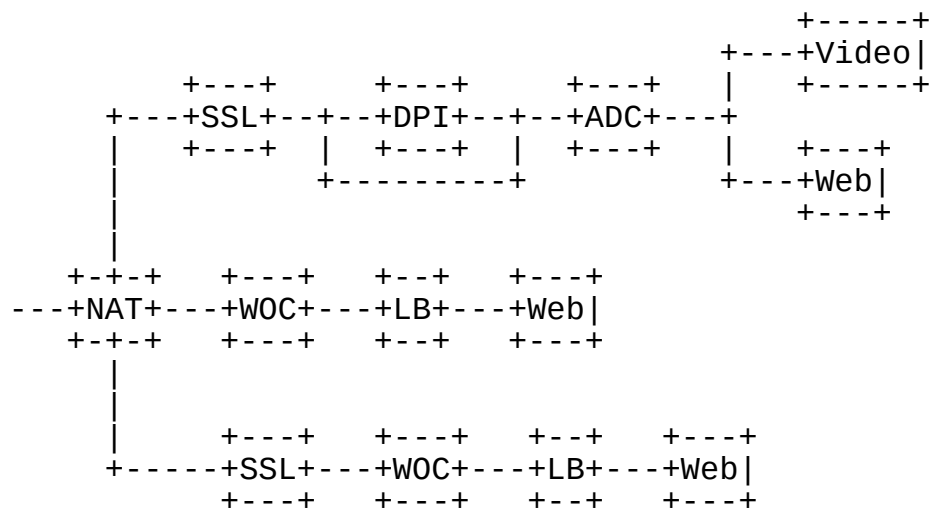

Figure 1 : Service function chains created by Enterprise B.

Suppose initially Enterprise B wants to support two user groups. One group includes all its employees. The other group is for visitors. The two user groups have distinctive service function requirements. Therefore Enterprise B has to create two SFCs out of its outsourced enterprise network. The first service chain, designed for its employees, will force traffic flows to go through NAT (Network Address Translation), SSL (Secure Socket layer)/TLS (Transport Layer Security), DPI (Deep Packet Inspection) if necessary, ADC (Application Delivery Controller), and various servers as shown in Fig.1. In the SFC, NAT, TLS, and DPI provide strong firewall service while ADC conducts service routing and load balance. The second SFC, designed for guest visitors, will go through NAT, WOC (Web Optimization Control), LB (Load Balancer), and web servers as shown in Fig.1. Here NAT provides limited firewall function with access

control. WOC and LB are designed to optimize server efficiency. Enterprise B will create these two service chains as overlays over its outsourced enterprise network. Because the underlying service chain has a mesh topology with all different service function nodes, Enterprise B can create the two service chains very fast with minimal efforts.

Suppose Enterprise B later wants to add another user group for one of its customers, called Customer C, it can do so easily by adding another service chain which may include NAT, SSL/TLS, WOC, and LB as shown in Fig.1.

Each user group is a tenant for Enterprise B. Therefore Enterprise B needs to assign an ID for each tenant so that it can differentiate traffic streams for the three different tenants. Each Id needs to be unique for Enterprise B.

Customer C may be an enterprise that has many departments who want to access the resources available at Enterprise B's network. Customer C is given full control of the service chain created for it. Customer C may then create a service chain and an ID for each department that needs access.

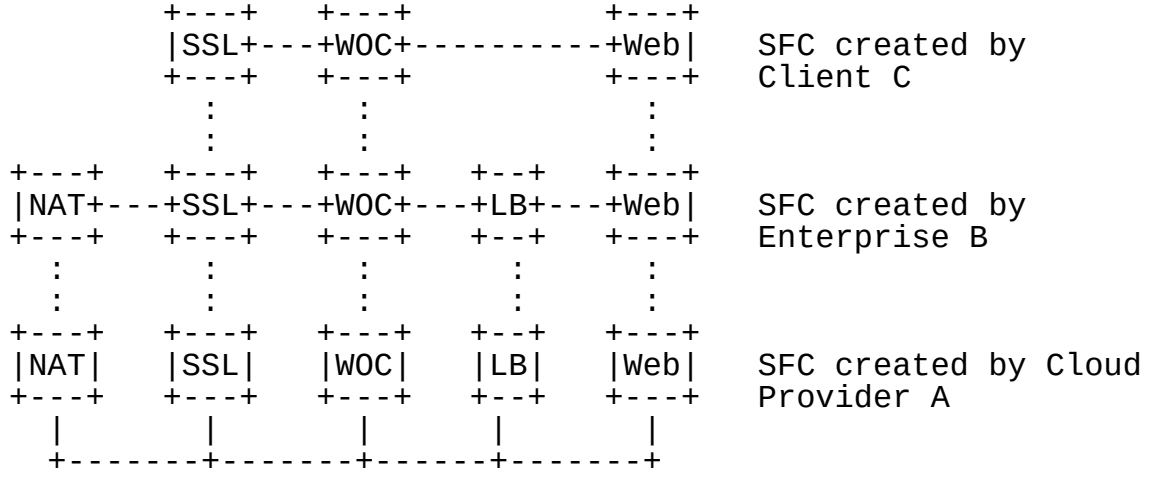

Figure 2 : Recursive service function chain structure.

The above structure clearly leads to a recursive service relationship as shown in Fig.2 where dot lines show mapping relationship and dash lines are service chains (For clearness, only one service chain is shown for each level.). Cloud Provider A provides the first level SFC that includes a customized topology and generic service function nodes. Enterprise B provides the second

level SFC which includes three customized SFCs. Customer C builds the third level SFCs for several departments over the SFC created by Enterprise B. As we mentioned before, this structure bears some similarity to the Carriers' Carriers concept defined in RFC 4364[1].

4. Analysis

One of the key issues introduced by the hierarchy of recursive SFC relationship is the relationship between different levels. There are two types of relationship that can be envisioned. The first one is called opaque relationship where the lower level is agnostic of the SFCs created by upper levels. Therefore all the service functions created by an upper level will be implemented and enforced at the upper level SFC modules while the lower level modules are completely unaware. When traffic arrives at a lower level module, the module processes the incoming traffic based on its service function requirements and de-multiplexes the traffic to the right upper level module using the IDs it assigned. The lower level module does not execute the service functions of upper level. The upper level applies different service functions based on the IDs it assigned. In this case, the upper level module does not have to be the same type as the lower level module (e.g. the lower level may be a NAT function while the upper level may be SSL function). But the upper level module will be based on the output of lower level module. For example, traffic that has been filtered by lower level cannot be recovered by upper level. This is why it is called opaque.

The other type is transparent relationship where service functions defined by upper level may require collaboration from lower level. For example, Enterprise B may inform Cloud Provider A about the SFCs it has created and ask Cloud Provider A to help implement flows belonging to different SFCs. When traffic arrives at Cloud Provider A, it will identify traffic flows using both the ID it assigned and the ID assigned by upper level as a concatenated ID and then apply associated service functions. The traffic stream will not be de-multiplexed to upper level. In this case, upper level functions inherit properties from lower level functions. They are also constrained by the functions available from lower level. However the upper level can create new properties such as new firewall rules as long as it doesn't violate the constraint posed by the lower level. Whenever service functions at lower level are changed, upper level service functions will also be changed. However changes made to the upper level may not apply to lower level. Fig.2 is an example of the transparent relationship.

In both cases, the upper level and lower level represent different authorities. Cloud Provider A decides the mesh service chain while

Enterprise B decides the three linear service chains for its three tenants. This is key feature of recursive service function chaining. In practice, a tenant is more likely to retain some functions as opaque (e.g. encryption function) and some functions as transparent (e.g. LB).

The above discussions show some special properties unique to recursive service chain. It is necessary to investigate how these properties can be supported using existing protocols, proposed SFC mechanisms, various other mechanisms, or even new proposals.

5. IANA Considerations

It is recommended that IANA assign a port in UDP and another port number in TCP to identify the existing of SFLs in Layer 5. The top level SFL of a SFL stack can use all existing port number assignments to identify various applications.

6. References

[1] E. Rosen and Y. Rekhter, “BGP/MPLS IP Virtual Private Networks (VPNs),” IETF RFC 4364, February, 2006.

Authors’ Addresses

Changcheng Huang
Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive
Ottawa, ON K1S 5B6
Canada
Email: huang@sce.carleton.ca

Jiafeng Zhu
Huawei Technologies Inc
Santa Clara, CA
US
Email: Jiafeng.zhu@huawei.com

Peng He
Ciena Corp
Email: phe@ciena.com