Service Function Chaining H. Yan
Internet Draft Tsinghua University
Intended status: Informational Y. Li
Expires: January 2018 Tsinghua University
H. Sun
Huawei Technologies
C. Xiong
Huawei Technologies
D. Jin
Tsinghua University
July 28, 2017

A Service Chain Aggregation Architecture
draft-ietf-sfc-aggregation-00.txt

Abstract

Service chains specifying the ordered sequences of network functions according to network policies requested, play a very important role in improving network performance in most networks. With the introduction of Software-Defined Networking (SDN) and Network Function Virtualization (NFV) technologies, service chains can be immediately deployed in more effective ways. Since different customers or operators would request multiple service chains in the same network but different administrative domains, and the multiple service chains may apply to the same network connection, we propose a service chain aggregation architecture that has ability to effectively aggregate them before the deployment. In this architecture, we determine the aggregating order of service chains according to different conditions verifying the aggregation effectiveness. The benefits of our architecture are 1) security, 2) network resource (e.g., flow entries in switches) savings and 3) scalability.

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 January 28, 2009.

Table of Contents

1. Introduction

Network policies can specify how network works and implement specific network functions. In most networks, operators usually manage the network via configuring the policies. For example, in LTE systems, the policy control and charging rules module is responsible for making the policy decision and operators create or update the policies via this module. As the demands of business scenarios become more diverse, customers have the need to customize the network services through network policies. However, it is not easy to meet with such ability because of the network inflexibility. For example, network

resources (e.g., network devices) are mostly fixed or statically configured in the underlying networks, and thus cannot be flexibly controlled.

To address this problem, Software-Defined Networking [RFC7426] and Network Function Virtualization (NFV) [etsi_gs_nfv_003], as two most promising technologies, are introduced for future network design. SDN decouples the control plane from forwarding plane and enables the logically centralized control; while NFV utilizes the virtualized technologies to implement the required network functions in generic servers instead of the proprietary network devices. By combining with these two technologies, it offers an opportunity to open the ability that applies for required network policies on demand to upper-layer applications.

Service chain is one of most important types of network policies in most networks [RFC 7665]. It consists of a set of ordered network functions (e.g., firewall, counter, deep packet inspection (DPI)) and can implement required network services. For example, Figure 1 shows a typical service chain in Evolved Packet Core (EPC). In this chain, optimizer can automatically adjust the packet format in order to match different mobile devices in real time, and firewall is provided to protect against attacks from external network. The deployment of this chain in the network is beneficial to improve user experience on the consumption of online contents via mobile devices.

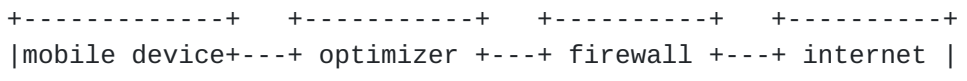

Figure 1: A typical service chain in EPC

With the advantages of SDN and NFV, service chains can be flexibly configured and deployed to the network. Importantly, it is usual to simultaneously request multiple chains for the same network connection or application. To illustrate this, we take Figure 2 for example. Chain 1 is defined to monitor the information of traffic rate; while Chain 2 specifies a DPI to inspect the packets' information. We can observe that Chain 1 does not influence Chain 2 and these two chains can be deployed at the same time. However, there is another case that for the same connection some service chains may have conflicting behaviors. Thus, direct deployment may influence other network functions and network performance. For example, Figure 3 shows two conflicting service chains for the same connection. Chain 1 specifies a firewall that drops the packets with the destination address of ip1; while Chain 2 has a redirector that modify the destination address of packets from ip1 to ip2. Intuitively, these chains must occur conflict since the processing behaviors for the same packets are contradictory (drop vs modify). Especially, if more service chains are requested for the same connection, it becomes more difficult and complex to verify their behaviors. To deal with it, we need to effectively aggregate these service chains without influencing original network functions. [chaithan_pga_sigcomm] studied the composition of graph-based policies. However, they do not give the complete solution on how to determine the order of aggregations.

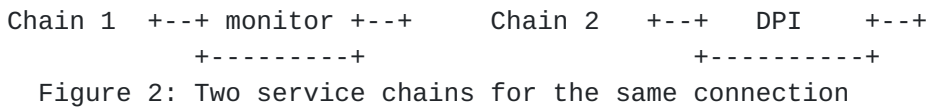

Figure 2: Two service chains for the same connection

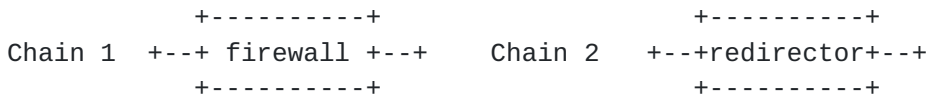

Figure 3: Two conflicting service chains for the same connection

In this document, we propose a novel architecture to support the aggregation of service chains. This architecture verifies the effectiveness of aggregating two service chains for the same source-destination pair and then determines final aggregation order before deployment.

The advantages of the architecture are summarized as follows: 1) it avoids the waste of resources like flow entries. For example, two service chains in Figure 2 can be aggregated as the form in Figure 4, which reduces the number of flow entries in the switch. 2) it enhances network security by detecting the conflicts; 3) it enables highly effective management for service-chain-based policies, and operators can easily update or add new requests.

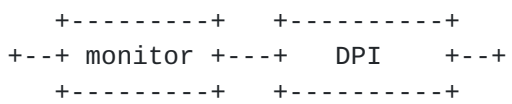

Figure 4: Aggregating two service chains in Figure 2 into a new one

2. Design Objective

Our proposed architecture is to automatically and effectively aggregate the service chains in SDN/NFV networks. The aggregation can not only avoid the behavior conflicts of service chains for the same connection, but also optimize network resources. In future networks, many applications rely on deploying different service chains. Therefore, it is necessary to design an aggregation architecture to reliably and intelligently implement them.

3. Requirements and Terminology

3.1. Requirements

The future network architecture has to support the separation of the control and forwarding plane. The controller in the control plane can globally arrange the network resources (e.g., bandwidth, forwarding rules, network functions or middlebox) in the forwarding plane.

Meanwhile, it needs to support the NFV capability. Network functions can be implemented by means of the virtualized technologies used in the generic sever. Thus, controller can flexibly deploy required network functions in the underlying network.

3.2. Definition of Terms

Network service: This consists of one or more network functions, which

is often provided by the operators.

Service chain: This defines an ordered set of virtual network functions. For example, firewall can be seen as a virtual network function.

Network function: It is responsible for processing the specific packets. A network function can be implemented as a virtual element in a generic sever or be embedded in a proprietary device.

4. The Architecture of Service Chain Aggregation

In this section, we introduce our architecture of service chain aggregation and illustrate how to use it to realize the aggregation of service chains.

4.1. Architecture Overview

Figure 5 shows the architecture of service chain aggregation. It consists of four major modules: application/management platform, aggregator, selector and controller.

Application/management platform is used for customers or operators to submit their requests of service chains. In addition, this module supports to query the information of aggregated service chains in the network, which is beneficial of operators to effectively manage the network.

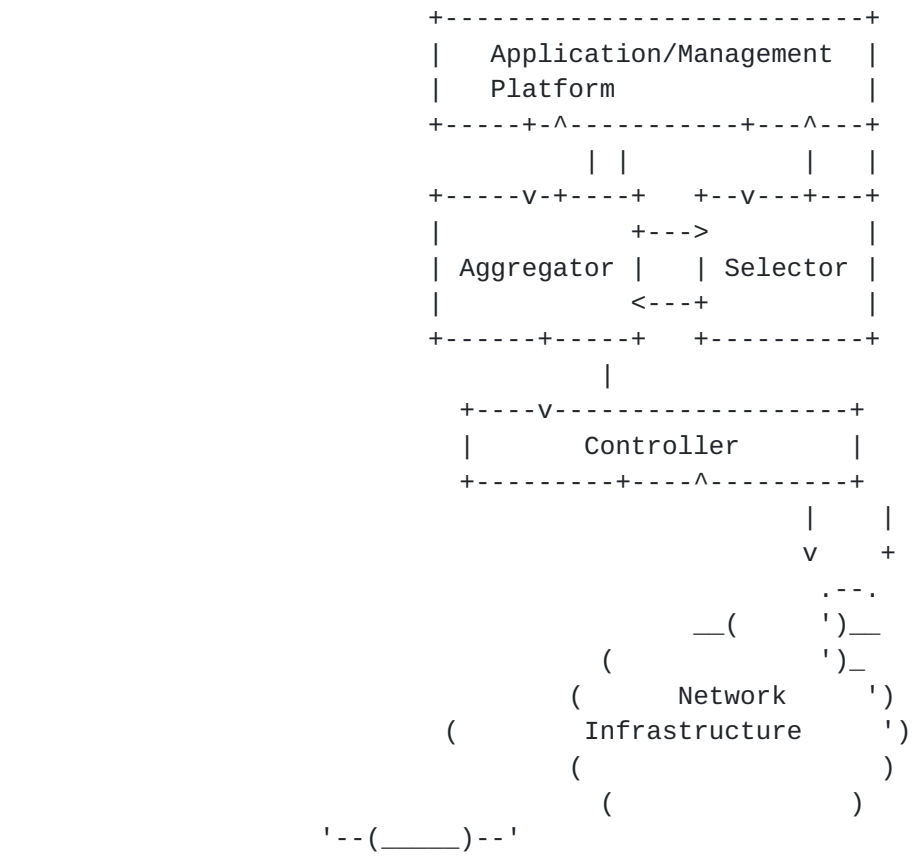

Figure 5: The architecture of service chain aggregation

Aggregator is a core module that aggregates the service chains. For service chains serving the same connection, this module analyzes the function of two chains and decides how to aggregate them.

Selector is used to select an aggregation order when it receives the request from the aggregator. Some default rules can be defined in the module. These rules define the placing order of network functions. In addition, operator can configure new rules via application/management platform.

Controller is responsible for converting the chains into the low-level configurations that implement specific network services. These configurations may be the forwarding rules that specify the packets to go through the prescribed forwarding elements, creating virtual network functions in the specific servers and so on.

4.2. Aggregation Mechanism

The key to aggregate service chains lies in classifying the possible aggregation scenarios. When aggregating two service chains, we intuitively deduce three possible results: the aggregation failure, one definite order and two optional orders. Specifically, when the behavior of newly requested service chain conflicts with the deployed one, the aggregation fails. For one definite order, it occurs when one of two aggregate orders would influence the original function of each chain. Finally, we can obtain two optional aggregation results if any one of them is fine.

The detailed process in aggregator is summarized as follows. The aggregator first checks whether the newly requested chain serves a new connection that has not been deployed a chain before. If this is the case, the new chain can be directly deployed without aggregation via controller. Otherwise, it will perform the aggregation process. During this process, it examines which condition of aggregation can be satisfied. If the same packets are processed by two chains but the corresponding behaviors conflict, the aggregator will notify the application/management platform the conflicting information. If no conflict is detected, it needs to check whether each function of two chains can be implemented after aggregation. We choose one order that can simultaneously satisfy both functions. Specifically, we estimate that an aggregation order is effective if it satisfies the following condition. The packets matched with the behavior of the first chain are consistent with the ones after its process, which are also matched with the behavior of second chain. If two aggregation orders are fine, the aggregator module will request the selector to determine one. Finally, the aggregation result will be stored in the database of aggregator and then sent to the controller.

In addition, the aggregator provides the service interface for application/management platform to request the information of aggregated service chains.

5. Examples of Service Chain Aggregation

In this section, we give some typical examples to illustrate how this architecture addresses with the newly requested service chains and guarantees the network reliability.

5.1. Verifying Conflicting Service Chains

Consider there are two administrative domains in the SDN/NFV-based network. To address with the abnormal traffic, two operators use different policies for the same connection. Operator 1 requests a service chain containing a firewall that filters the packet from source address ip1, but Operator 2 designs another service chain consisting of a remarker that remarks the same packets. We assume that there was no chain in this connection before. Figure 6 shows these two chains.

Figure 6: Two service chains in example 1

When one of these chains is requested via application/management platform, it will be sent to aggregator. Since no chain exists in this connection, aggregator would notify the controller to deploy this chain. When another chain is sent to aggregator, this module will check that the behaviors of two chains (drop vs remark) conflict. Then, it immediately notifies the application/management platform the information of the conflict.

5.2. Only One Aggregation Order of Service Chains

Consider an operator has deployed a service chain to count the number of packets with destination address of ip1 in some connection. When number of such packets drastically increases, the operator wants to request a new service chain to modify destination address of these packet to a new IP address. Figure 7 shows these two chains.

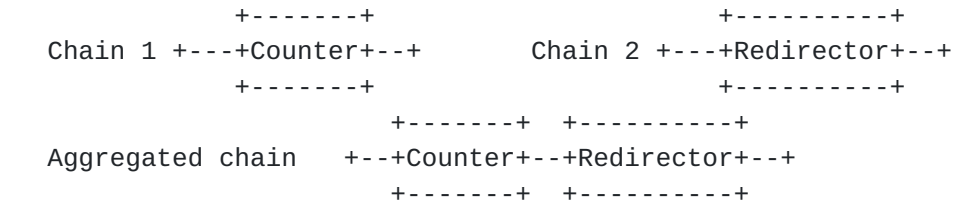

Figure 7: Two service chains and its aggregation result in example 2

Since Chain 1 has deployed into the underlying network, the system would find this information and then attempt to aggregate Chain 1 and 2. Specifically, when Chain 2 is forwarded to the aggregator, this module would check whether their behaviors are contradictory. After verifying no conflict exists, the aggregator starts to determine its aggregation order. If Chain 1 is placed after the Chain 2, no packets with the destination address of ip1 are counted, which influences the network service of Chain 1. However, if the aggregation order is opposite, the network services of both chains can be met with. This is because that two chains process the same packets and the counter

does not change the destination address of packets. Finally, the aggregation result is sent to controller for deployment.

5.3. Multiple Aggregation Order of Service Chains

Consider an operator has deployed a service chain to count the number of packets with destination address of ip1 in some connection. After a short time, the operator wants to request a new service chain to inspect the information of the same packets. In addition, the rule in selector prescribes that DPI should be placed before Counter. Figure 8 shows these two chains.

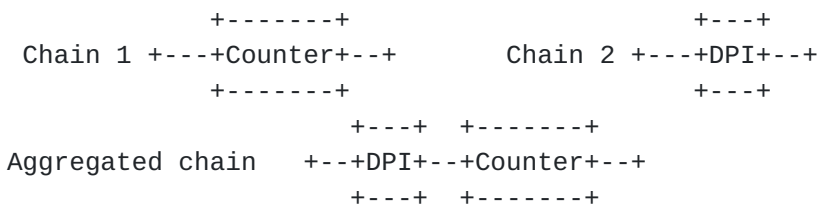

Figure 8: Two service chains and its aggregation result in example 3

Like the initial process in example 2, aggregator starts to determine the aggregation order of Chain 1 and Chain 2. Since no matter how the chains aggregate, their original functions can be fulfilled. Therefore, the aggregator sends the request to selector. Then, the selector determines the final order shown in Figure 8 according to the predefined rules and returns it to the aggregator.

6. Summary

This document proposes an architecture for service chain aggregation based on the SDN/NFV network. This architecture utilizes the analysis of possible aggregation results, thus assuring its effectiveness. Meanwhile, we give some typical examples to illustrate how this architecture works. Our architecture can not only meet with required network services but also assure network reliability. Because future Evolved Packet Core would use SDN and NFV technologies, this architecture is also applicable in this environment.

7. Security Considerations

Security issues due to aggregating the service chains across different administrative domain are an aspect for further study.

8. IANA Considerations

This draft does not have any IANA considerations.

9. Informative References

[etsi_gs_nfv_003] ETSI NFV ISG, "Network Functions Virtualization (NFV); Terminology for Main Concepts in NFV", ETSI GS NFV 003 V1.2.1 NFV 003, December 2014, <http://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.02.01_60/gs_NFV003v010201p.pdf>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <http://www.rfc-editor.org/info/rfc7665>.

[RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <http://www.rfc-editor.org/info/rfc7426>.

[chaithan_pga_sigcomm] Chaithan Prakash, Jeongkeun Lee, Yoshio Turner, Joon-Myung Kang, Aditya Akella, Sujata Banerjee, Charles Clark, Yadi Ma, Puneet Sharma, and Ying Zhang. 2015. PGA: Using Graphs to Express and Automatically Reconcile Network Policies. SIGCOM 2015, 29-42.

10. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Huan Yan
Tsinghua University
yanh14@mails.tsinghua.edu.cn

Yong Li
Tsinghua University
liyong07@tsinghua.edu.cn

Haiyang Sun
Huawei Technologies
sunhaiyang3@huawei.com

Chunshan Xiong
Huawei Technologies
sam.xiongchunshan@huawei.com

Depeng Jin
Tsinghua University
jindp@tsinghua.edu.cn