Computer Networks 92 (2015) 396–407

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Enabling network function combination via service chain instantiation

Guozhen Cheng, Hongchang Chen*, Hongchao Hu, Zhiming Wang, Julong Lan

National Digital Switching System Engineering and Technological R&D Center, Zhengzhou 450002, China

ARTICLE INFO

Article history:
Received 31 October 2014
Revised 25 July 2015
Accepted 11 September 2015
Available online 26 September 2015

Keywords:
Software-defined networking
Network function virtualization
Function combination
Service chain

ABSTRACT

Isolated network functions (also known as middleboxes) are difficult and costly to manage in a cooperative fashion due to their hardware-based implementation and proprietary interfaces. The integration of software-defined networking (SDN) and network functions virtualization (NFV) is promising to address this challenging issue. However, an efficient framework is needed to provide cooperative control of network function instances. To this end, a service chain instantiation framework based on NFV and SDN is proposed in this paper. There are three unique contributions in this work. *First*, the network functions are featured with a new abstraction, called atomic function, which defines the public features of network functions while the core details are hidden. A description-language is utilized to help service providers to develop various instances of an atomic function. *Second*, we propose an implementation of service chain consisting of a sequence of atomic functions with order constraints. It is instantiated by optimally selecting different function instances over the network. We formulate this service chain instantiation as an integer linear programing problem, with a simulated annealing solver to approach the optimal solution. *Third*, we implement a proof-of-concept for service chain, Matchmaker, atop the SDN controller. The experimental results demonstrate that Matchmaker can manage network functions in an efficient and scalable way.

1. Introduction

Many hardware-based network appliances (i.e., middleboxes) which examine and modify packets and flows in sophisticated ways are comprise the closed data plane in the current Internet. They support a diverse set of useful functions ranging from security (firewalls), traffic shaping (load balancers), and network optimizer (caching proxies), and so on. Although middleboxes play a critical role in ensuring security, improving performance, and providing novel functions, they are costly and difficult to manage in an optimal fashion or to scale up and down with shifting demands due to their hardware-based implementation and proprietary interfaces.

Recently, network function virtualization (NFV) [1] has been proposed to replace dedicated hardware-based middleboxes with software-based alternatives running on inexpensive, commodity hardware (e.g., x86 server). At the same time, software-defined networking (SDN) is being used to steer flows through appropriate network functions to manage the network and enforce policies [2–4].

The integration of "SDN+NFV" eases the network innovation via outsourcing network functions [5], not only for infrastructure providers but also enabled third-party service providers like Google and Facebook could participate in developing the network functions and deploying their instances in the network. It is anticipated that, this architecture will inevitably become popular for handling network functions in the near future. However, the current SDN lacks an appropriate mechanism that glues those functions together

* Corresponding author. Tel.: +8615137172806.
E-mail addresses: guozhencheng@hotmail.com (G. Cheng), chc@ndsc.com.cn (H. Chen), huhongchao@gmail.com (H. Hu), wangzm05@gmail.com (Z. Wang), ndscljl@163.com (J. Lan).

http://dx.doi.org/10.1016/j.comnet.2015.09.015
1389-1286/

to serve dedicated flows with a tight service level agreement (SLA).

In legacy networks, new functions may be embedded in end-hosts or added in switches and routers, even in middleboxes [6] when needed. It is impossible for network operators to manage them in a cooperative way due to the lack of the global information about them.

In "SDN+NFV", each authorized service provider can readily develop various software-based function instances in the servers. SDN switches can steer traffic flows through those instances so that they could process flows on demand. The SDN controller manages network switches and servers. Accordingly, various instances from different service providers could coordinate with each other in a more optimal way. There are three primary features for "SDN+NFV" regarding function innovations. *First*, network functions will be increasingly decomposed into fine-grained ones.

Second, any service provider could provide their function instances. It is possible that one network function has more than one instance in the network. To implement NFV, many commodity servers can be deployed to support function instances. In the extreme case, each switch could connect a server. All servers are managed by SDN controller.

Finally, several primitive network functions can be combined into a complex pipeline based on the operator's policy so that given flow could flow through the pipeline. This process should be conducted carefully to fully utilize the resource. For example, we should distribute flows among different instances of a function for load balance.

However, simultaneously achieving all three features requires a framework that provides efficient and coordinated control of function instances over the network. First, it needs a function description-language to depict function information so that the public features of network functions could be defined and used to conduct instance developers, but the core detailed features are hidden. Second, it should provide a mechanism to orchestrate instances on demand.

To this end, a service chain instantiation framework (SCI) based on "SDN+NFV" is proposed to address the aforementioned challenges. The main contributions of this paper are

- The network functions are firstly abstracted as fine-grained building blocks, called atomic function (AFs). A description language is then utilized to define the atomic function and publish its public features (i.e., types, parameters and SLAs) so that the AF providers (AFP) can be conducted to implement it.
- A new mechanism is proposed to combine AF instances into service chains on demands. A service chain is defined as a customized function process chain created on demands of an integrated application. We formulate service chain instantiation into an integer linear programming problem. As the problem is a NP-hard, a simulated annealing algorithm (SA) is developed to approach the optimal solution to the problem.
- Taking SA solver as the core component, we design and develop a prototype implementation of SCI atop the SDN controller, called *Matchmaker*.

The remainder of this paper is organized as follows. Section 2 gives the related works. Section 3 discusses the related concepts for SCI. Section 4 presents the SCI problem. Section 5 indicates the algorithm for SCI problem. Section 6 presents design and implementation of Matchmaker. Section 7 presents the experimental evaluation of our Matchmaker and a use case. Finally, Section 8 concludes this paper.

2. Related work

Middleboxes have become the most important forms to extend core network functions in the legacy Internet for external minority enablers other than infrastructure providers. For example, we can deploy a load balancer on a flow congestion point [6]. However, the current hardware-based middleboxes are costly, difficult to manage, and impossible to scale up and down with shifting demands.

Network function structure: New framework in xOMB [7] and CoMb [8] argue for extensible middleboxes on the commodity hardware. CoMb exploits the middlebox structure to consolidate heterogeneous middleboxes onto a commodity hardware. FreeFlow [9] gives a common structure of middlebox which define three types of states to be exposed. Our work can benefit from these, especially in the context of function interfaces. But this paper describes network functions by a unified function description language.

The Integration of SDN/NFV and middlebox: Recently SDN has been employed to propose software-based programmable middleboxes. Qazi et al. [10] provide a SIMPLE policy enforcement layer based on SDN to reduce manual effort and operator expertise for middleboxes, but it takes each middlebox as a black box and much contextual information will be missing. Gember et al. [11] implement a software-defined middlebox networking framework to simplify the management of complex, diverse functionalities. Further OpenNF [12] provides efficient, cooperative control for reallocation of flows across network functions (i.e., middleboxes). It presents the combined benefits of NFV and SDN. However, those promising works did not deal with how to combine disheveled network functions to satisfy special demands. ClickOS [13] provides an implementation of virtual network functions. It could create small virtual machines (5 MB) which boot quickly and add little delay. Each VM could host one middlebox. Network policies typically require packets to go through a sequence of middleboxes (e.g., firewall + IDS + NAT + proxy), but ClickOS does not provide a mechanism to generate such sequence in an optimal ways.

Application composition in SDN controller: Recently, several works are provided to design SDN northbound APIs to implement different functions, called control applications. Frenetic [15] and Pyretic [16] give a composing language to manually arrange the control applications, but they present more stress on the rapid innovations, and there is no smart decision-making mechanism for application selections. FRESCO [17] is a click-inspired programing framework for network security applications. We contrast our work with FRESCO: both systems support composition of modules. Matchmaker supports composition of modules in higher-level policy. In other words, Frenetic, Pyretic, FRESCO and other prior works that we know of, aim at better ways to get the network to do what you want it to do; Matchmaker, in contrast, is aimed at deciding what you want it to do. As a