Demo: Chaining of Segment Routing aware and unaware Service Functions

Ahmed Abdelsalam
Gran Sasso Science Institute

***Abstract*—Segment Routing (SR) is a source routing paradigm that can benefit from both MPLS and IPv6 data planes to steer traffic through a set of nodes. It provides a simple and scalable way to support Service Function Chaining (SFC). In this demo, we propose an NFV architecture based on SR and implemented in Linux environment. It allows chaining of both SR-aware and SR-unaware Service Functions (SFs). In order to include SR-unaware SFs into SR SFC, we use our SR proxy implementation: srext, a Linux kernel module that handles the processing of SR information in behalf of the SR-unaware SFs. As SR-aware SFs, we use two of our implementation; SERA and SR-aware snort. SERA is a SEgment Routing Aware Firewall, which extends the Linux iptables firewall, and capable of applying the iptables rules to the inner packet of SR encapsulated traffic. SR-aware snort is an extended version of snort that can apply snort rules directly to inner packet of SR encapsulated traffic. We show the interoperability between SR-aware and SR-unaware SFs by including both of them within the same SFC.**

***Index Terms*—Service Function Chaining, Network Function Virtualization, Segment Routing, Linux networking**

I. Introduction

Telecommunication networks infrastructures are evolving at a rate rarely seen since the transformation from analog to digital [1]. Network functions virtualization (NFV) offers an agile way to design and deploy networking service [2]. In an NFV infrastructure, network functions are decoupled from proprietary hardware appliances and moved to virtual servers so they can run in software modules called Virtual Network functions (VNFs), which are sometimes referred to as Service Functions (SFs). This dramatically reduces both capital expenditures (CAPEX) and operating expenses (OPEX) [3]. A set of these VNFs (SFs), which can be arbitrarily located in a distributed virtualization infrastructure, are often required to deliver an end-to-end service, hence Service Function Chaining (SFC) comes into play.

SFC denotes the process of forwarding packets through the sequence of SFs [4]. It requires a steering mechanism to force packets to go through SFs. These steering mechanisms often require inserting a new header into packets, which carries the path information. Network Service Header (NSH) and IPv6 Segment Routing header (SRH) are two examples of those headers. NSH has been proposed by the IETF SFC Working Group to support the encapsulation of packets with a header that specifies the sequence of SFs to be crossed [5]. Using NSH requires creating a state (per each NFV chain) in the network fabric, which doesn't make it the preferred solution in the recent era of networking, where everything is going towards stateless and simplicity. On the contrary, using SRH for SFC doesn't have the need for those state information.

In this demo, we consider the use of the Segment Routing (SR) architecture to support SFC. SR is a new network architecture that leverages the source routing paradigm [6]. It allows to steer packets through an ordered list of nodes, which are refereed to as segments. SR can be instantiated over both MPLS (SR-MPLS) and IPv6 (SRv6) data planes. SRv6 defines a new IPv6 Routing type, named SRH. It allows including a list of segments in the IPv6 packet header [7]. Each A segment is encoded as an IPv6 address and represents function to be called at a specific location in the network. SR enables SFC in a simple and scalable manner, by associating each SF with a segment. Such segments are combined together in a segment list to achieve SFC.

II. SR-aware vs SR-unaware SFs

SFs can be categorized into two types, depending on their ability to properly process SR encapsulated packets. These are respectively named SR-aware and SR-unaware SFs [8]. An SR-aware SF is able to correctly process SR-encapsulated packets it receives, which imply being able to process the original packet despite the fact that it has been encapsulated within a SR packet, but also being able to process the SRH. On the contrary, An SR-unaware SF is not able to correctly process the SR-encapsulated it receives. It may either drop the traffic or take erroneous decisions due to the unrecognized SR information. In order to include SR-unware SFs in an SR SC policy, it is thus required to remove the SR information as well as any other encapsulation header before the SF receives the packet, or to alter it in such a way that the SF can correctly process the packet. SR proxy is an entity, separate from the service, that performs these modifications and handle the SR processing on behalf of a SR-unaware service. Srext [9] is a Linux kernel module providing advanced SR functions. It supports different SR proxy behaviours detailed in [8].

III. SR/SFC Testbed

In order to showcase the SFC of SR-aware and SR-unaware SFs, we built the testbed shown in Figure 1, which consists of six nodes (R1-R6), implemented as Linux VMs and represent our SR domain. This SR domain is used to connect two branches (BR1 and BR2) of an enterprise to an external network (Ext). All nodes, except R4, support SRv6. Nodes R1 and R6 respectively represent the ingress and egress nodes, while nodes R2, R3 and R5 are used as NFV nodes of our

SRv6 based SFC scenario. Node R4 is a normal IPv6 Linux router. Service Functions (F1-F3), Branches (BR1 and BR2), and external network (Ext) are deployed as Linux network namespaces. F1 is an SR-aware Linux iptables firewall, F2 is SR-unaware snort, and F3 is an extended SR-aware version of snort. Nodes R1, and R4-R6 are running kernel 4.14 and have iproute2 v4.14 installed. Node R2 is running compiled Linux kernel 4.15-rc2 with SRv6 enabled and SERA firewall included [10]. Node R3 has the srext [9] kernel module installed. The links between any two nodes Rx and Ry are assigned IPv6 addresses in the form `fc00:xy::x/64` and `fc00:xy::y/64`. For example, the two interfaces of the link between R1 and R2 are assigned the addresses `fc00:12::1/64` and `fc00:12::2/64`. Each node owns an IPv6 prefix to be used for SRv6 local SID allocation, which is in the form `fc00:n::/64`, where `n` represents the node number. As an example, R2 owns the IPv6 prefix `fc00:2::/64`. SFs are instantiated on an SR SID of form `fc00:n::fk:/112` where `n` represents the node hosting the SF and `k` is the SF number. For example, F1 which is running in node R2 is given the prefix `fc00:2::f1:/112`. BR1, BR2, and Ext are respectively assigned the IPv6 prefixes `fc00:b1::/64`, `fc00:b2::/64`, and `fc00:e::/64`.

IV. SR/SFC POLICIES

The testbed in Figure 1 supports two different path, with different bandwidth and security guarantees, towards *Ext*. Path p_1 ($R_1 \rightarrow R_4 \rightarrow R_5 \rightarrow R_6$) provides high bandwidth. Path p_2 ($R_1 \rightarrow R_2 \rightarrow R_3 \rightarrow R_6$) has lower bandwidth, but more security guarantees. Going through p_1 implies crossing F1 and F2. The same way, going through p_2 implies crossing F3. BR1 and BR2 have different traffic requirements; BR2 traffic is very delay-sensitive, while BR1 traffic is highly confidential, but less delay sensitive. We exploit p_1 and p_2 to satisfy those traffic requirement. BR1 traffic is steered through p_1, and BR2 traffic is steered through p_2. At the ingress node (R1), we configured two different SR SFC policies (CP1 and CP2) that steer traffic through p_1 and p_1. Policy Based Routing (PBR) is used to classify traffic coming form BR1 and BR2, which respectively go through CP1 and CP2.

V. DEPLOYMENT AND TESTING

We built our testbed by using *VirtualBox* [11] as hypervisor and *Vagrant* [12] as VM manager. This makes it easy to replicate the demo on any commodity hardware. Scripts required to deploy the demo are open source and can be found at [13]. To verify the deployment of the demo, we use *iperf* [14] to generate traffic from BR1 and BR2. BR1 traffic should cross both F1 and F2. F1 is configured with iptables rules, which are applied by SERA firewall directly to inner packet of received SR traffic. F2 is an SR-unaware snort, which can't correctly processes SR packets. We used srext to remove SR encapsulation from packets before being handed to F2. The removed SR encapsulation is re-added again to packets after being processed. F3 is the only SF crossed by BR2 traffic. It's an SR-aware snort that can apply configured snort rules directly to inner packet. To make sure that BR1 and BR2 traffic follows the exact path in both upstream and downstream, we configure two SR SFC policies on the egress node (R6) for the reverse path. This guarantees that SFs get the traffic in both directions.

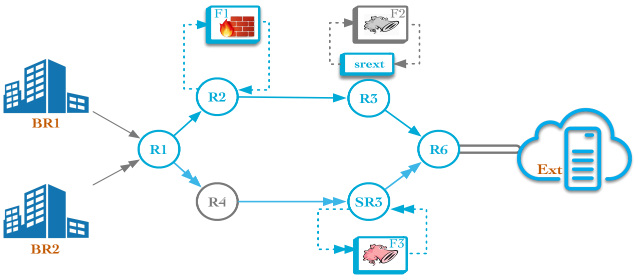

Fig. 1: Testbed for SR/SFC demo

VI. CONCLUSIONS

In this demo, we introduce a Linux NFV infrastructure that support SFC of both SR-aware and SR-unaware SFs. We used our SR proxy implementation (srext) to include those SR-unaware SFs in an SR SFC. As SR-aware SFs, we provided two implementations of SR-aware SFs. SR-aware and SR-unaware SFs have been included in the same SFC to show their inter-operability. We provided an open source implementation for the SR/SFC testbed been used.

REFERENCES

[1] B. Thekkedath, *Network Functions Virtualization For Dummies*. Wiley, 2016.

[2] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[3] R. Mijumbi et al., "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, 2015.

[4] J. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," Internet Requests for Comments, RFC Editor, RFC 7665, October 2015.

[5] P. Quinn, U. Elzur, and C. Pignataro, "Network Service Header (NSH)," Internet-Draft, November 2017. [Online]. Available: http://tools.ietf.org/html/draft-ietf-sfc-nsh

[6] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The Segment Routing Architecture," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–6.

[7] S. Previdi (ed.) et al., "IPv6 Segment Routing Header (SRH)," Internet-Draft, September 2016. [Online]. Available: http://tools.ietf.org/html/draft-ietf-6man-segment-routing-header-02

[8] F. Clad *et al.*, "Segment Routing for Service Chaining," Internet-Draft, October 2017. [Online]. Available: https://tools.ietf.org/html/draft-clad-spring-segment-routing-service-chaining-00

[9] "srext - a Linux kernel module implementing SRv6 Network Programming model," Web site. [Online]. Available: https://github.com/netgroup/SRv6-net-prog/

[10] "SERA - SEgment Routing Aware Firewall," Web site. [Online]. Available: https://github.com/SRouting/SERA

[11] "VirtualBox home page," Web site. [Online]. Available: http://www.virtualbox.org/

[12] "Vagrant home page," Web site. [Online]. Available: http://www.vagrantup.com/

[13] "SRv6 SFC demo," Web site. [Online]. Available: https://github.com/SRouting/sr-sfc-demo

[14] "iPerf - The ultimate speed test tool for TCP, UDP and SCTP," Web site. [Online]. Available: http://iperf.fr