Service Function Chaining
Internet-Draft
Intended status: Informational
Expires: September 22, 2016

D. Dolson
M. Marchetti
K. Larose
Sandvine

March 21, 2016

Efficient Patterns for Service Function Chaining within Network Function Virtualization Infrastructure

draft-dolson-sfc-nfv-patterns-00

Abstract

The document presents some considerations for efficiently deploying Service Function Chaining (SFC) within a Network Function Virtualization Infrastructure (NFVI).

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). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on September 22, 2016.

Copyright Notice

Table of Contents

1. Introduction

Service Function Chaining (SFC) is a technique for prescribing differentiated traffic forwarding policies. SFC is described in detail in the SFC architecture document [RFC7665], and is not repeated here.

Network Function Virtualization (NFV) is technology for deploying network forwarding software functions on an infrastructure providing generic compute and network resources. Such an infrastructure is termed NFVI [ETSI_NFV].

This document presents some efficient patterns for deploying SFC within an NFVI, in the hope of sharing good practices.

1.1. Requirements Language

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 [RFC2119].

2. Objectives

The patterns described in this document are designed to satisfy:

- Minimize latency by minimizing both the number of physical hops and number of queues each packet traversing a service chain must undergo.

- Minimize CPU processing by avoiding unnecessary software switching.

These objectives serve both to increase network performance that would be measured by end users and increase efficiency of NFVI by minimizing switching and CPU resources required to implement each chain.

3. Assumptions

We wish to discuss solutions that are available with current technology. In particular:

- the NFVI is to be built using only currently available commercial off-the-shelf (COTS) hardware;
- the infrastructure is to be built using currently available NFVI technology and hosting, in particular the ability of the NFV infrastructure to provide virtual compute resources with interfaces to virtual Ethernet LAN segments.

4. Patterns

The following sections describe design patterns for using SFC that we consider to be appropriate under the above NFV assumptions.

4.1. Use MAC-NSH to address functions

A common infrastructure model (e.g., OpenStack/Neutron) allows provisioning of virtual machines with interfaces on specified Ethernet segments. Although the physical implementation of an Ethernet segment is opaque to the tenants, all machines on a segment can send packets to one another by addressing the destination MAC address.

Single-Root I/O Virtualization (SR-IOV) is a technology that allows a single physical interface on a compute host to be split into multiple virtual interfaces such that each guest has a hardware interface to the network. When so configured, the physical interface hardware sorts incoming packets into queues for the distinct emulated interfaces according to destination MAC addresses. This technology removes any need for a software module to sort packets received from the physical interfaces into virtual interfaces of the guests. Thus, an extra queuing stage is avoided and no CPU switching resources are required.

The NVFI operator may choose to deploy simple switching, with hardware backplane and top-of-rack Ethernet/VLAN switches providing minimum latency. The operator may also deploy Ethernet-over-IP (e.g., VXLAN) when functions are remote. In any case, the virtual host is agnostic to the implementation.

Thus, by using SR-IOV and efficient zero-copy drivers, functions naturally address one another by MAC address on the layer-2 segment. In the optimal NFV scenario, software in one function queues packets to an outbound SR-IOV queue, hardware sends the packet on the physical interface, backplane or top-of-rack switches deliver the packet to the physical destination and SR-IOV destination queue. There need not be any software between functions, only Ethernet switches.

Note that to capture the benefit of using SR-IOV to avoid software switching, any L2-over-L3 (e.g., VXLAN) should be arranged for in hardware. The functions themselves can then behave the same whether virtual or physical network segments are used.

We conclude that carrying NSH packets directly within Ethernet frames is an efficient and natural way to implement Service Function Chaining. The Ethernet encapsulation of NSH is documented in [I-D.ietf-sfc-nsh].

4.2. Locate SFF with the SF

Service function chains are often drawn like this two-SF example in which a packet goes from ingress to Classifier, to SFF1, to SF1, back to SFF1, to SFF2, to SF2, back to SF2 and finally to egress:

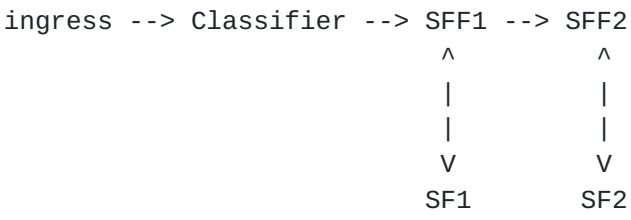

Figure 1: A conceptual service function chain

But in NFV infrastructure, if SFFs are considered distinct processing elements, the common reality is that each of the components are separated by Ethernet switching:

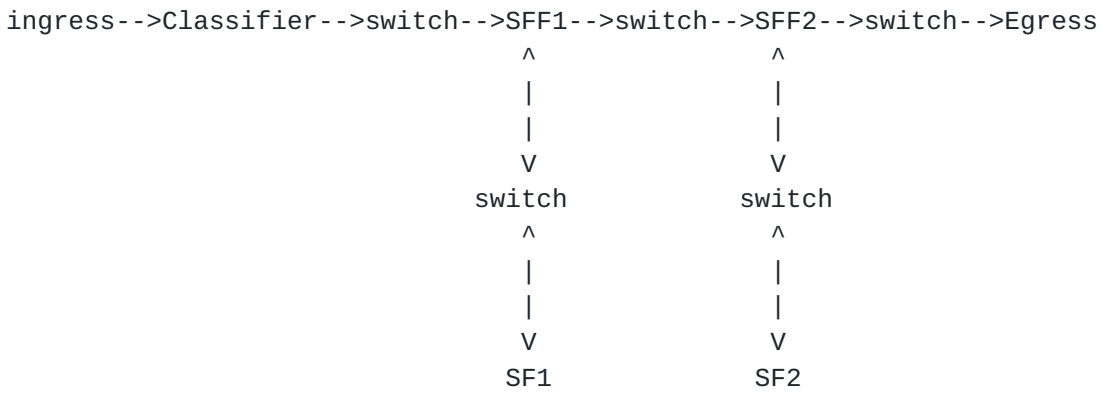

Figure 2: A service function chain showing switches

In an NFV environment, the "switch" blocks in Figure 2 can be implemented by Ethernet backplane, top-of-rack switching or (for remotely separated virtual machines) VXLAN virtual layer-2 network.

There is an optimization that can be made when all of the "switch" modules are the same Ethernet switching domain. In other words, when this is the logical topology of Figure 3:

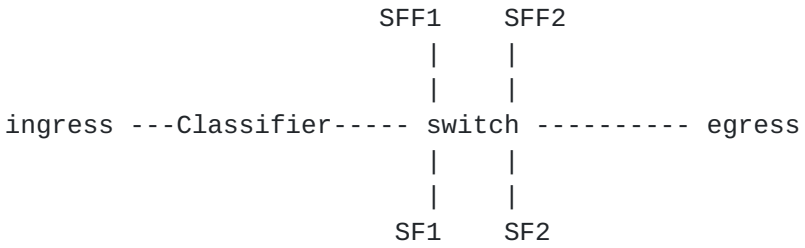

Figure 3: Service chain components attached to a layer-2 network

Notice that to implement the path of Figure 1 and Figure 2, a packet must pass through the switch seven times. It must also pass through each SFF twice, limiting the capacity of an SFF.

We can reduce the number of transits through the switch by placing SFF forwarding tables in the SF software module itself, like this:

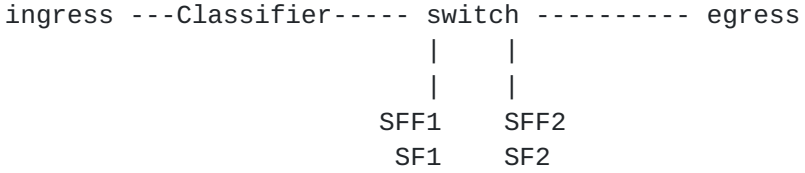

Figure 4: SFFs embedded in SF machines

In this case, a packet taking the path of Figure 1 only transits the switch three times. For minimal latency, an implementation can avoid queuing between the SF and attached SFF.

Although keeping the SFFs distinct from the SFs has an architectural purity, the purity has a big cost in switching throughput and corresponding cost in money and latency. Since each packet enters and exits each SFF twice, there could also be contention on the SFF interfaces.

There is additional control-plane burden to achieve this data-plane efficiency gain:

- SF software must implement SFF lookup functions. We feel this computation of next-hop and packet formatting can be done efficienty in software, but it does require support of the SFF feature in the software.

- There will be as many SFFs as SFs, each of which much have correct forwarding entries populated by the control plane during the orchestration of bringing an SF on line.

There are many ways to physically realize the logical switch entities in Figure 3 and Figure 4. The best case is a hardware Ethernet switch; considerations about how to optimally deploy functions are discussed elsewhere, e.g., in Network Function Virtualization Research Group (NFVRG). In any of these cases, reducing the number of passages though the switch will reduce latency and reduce operational cost.

4.3. Prefer NSH MD Type 2

The NSH draft [I-D.ietf-sfc-nsh] defines two options for metadata, types 1 and 2. Type 2 is more efficient when there is no metadata, and is the only choice supporting more than 128 bits of metadata.

Using the approaches described in Section 4.1 and Section 4.2, NSH packets are transported from one MAC endpoint to another via standard Ethernet switches, so there are no requirements on the NFV Infrastructure to support NSH.

Given that parsing the variable-length header poses no concerns for software, we feel these considerations make MD Type 2 the preferred choice for service chaining with NFV.

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

We are not aware of any additional security concerns raised by employing the methods discussed in this memo.

The MAC-NSH approach assumes security of (virtual) LAN segments.

Employing SFF functionality within Service Function software puts trust in that software, but SF functions must in any case be trusted to return packets to the correct path.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

7.2. Informative References

[ETSI_NFV] ETSI, "Network Functions Virtualization (NFV); Architectural Framework", October 2013, <http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf>.

[I-D.ietf-sfc-nsh] Quinn, P. and U. Elzur, "Network Service Header", draft-ietf-sfc-nsh-02 (work in progress), January 2016.

[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>.

Authors' Addresses

David Dolson
Sandvine
408 Albert Street
Waterloo, ON N2L 3V3
Canada

Phone: +1 519 880 2400
Email: ddolson@sandvine.com

Michael Marchetti
Sandvine
408 Albert Street
Waterloo, ON N2L 3V3
Canada

Phone: +1 519 880 2400
Email: mmarchetti@sandvine.com

Kyle Larose
Sandvine
408 Albert Street
Waterloo, ON N2L 3V3
Canada

Phone: +1 519 880 2400
Email: klarose@sandvine.com