This document reports on the experimentation with a combined Network Function Virtualization (NFV) orchestrator and Service Function Chain (SFC) Control Plane proof of concept prototype.

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

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Introduction

The goal of this report is to provide an insight into a Service Function Chain (SFC) control and Network Function Virtualization (NFV) orchestration proof of concept implementation and experimentation in multi-domain/technology environments.

The reported concept is based on the EU-FP7 UNIFY project’s approach \[I-D.unify-nfvrg-challenges\], \[UNIFY-GLOBECOM\], which defines joint compute and network virtualization and control interfaces \[I-D.unify-nfvrg-recursive-programming\].

The rest of the document is structured as follows: In Section 2 we define the related terminology, which is followed with the listing of our assumptions in Section 3. In Section 4 we derive our design from the SFC Control Plane framework. In Section 5 we outline our experimental network scenario. In Section 6 we analyze the gaps between our experimental design and the SFC Control Plane. Finally, in Section 7 we formulate some questions to the community based on our learnings.

2. Terms and Definitions

The reader should be familiar with the terms defined in \[RFC7498\] and \[RFC7665\].
3. Assumptions

We assume a Network Function Virtualization (NFV) environment, where Service Functions (SF) of a SFC will map as Virtualized Network Functions (VNFs).

We assume that both Classification and Service Function Forwarding (SFF) behavior can be expressed by SDN forwarding rules (match rules and actions); if a Classification or a SFF cannot be mapped to match and action pairs, then a corresponding control plane SF (VNF) is instantiated to analyze and tag the flows into SFP. (This is not different than an SF being SFC aware.)

We assume the existence of multiple SFC domains as a result of technology, administration or ownership considerations.

We assume that a boundary node can act as ingress/egress for Service Function Paths entering/leaving the SFC domain.

We assume dynamic establishment of SFC; static configuration is not considered in this document.

4. SFC Control Plane: An Experimental Design

We start with the reference architecture of [I-D.ietf-sfc-control-plane]:

[Page 3]
Let’s map the above architecture to Logical Data Plane Components, where classification / forwarding is controlled by SDN forwarding behavior control over interface "C" consumed by SFC Controller’s C1 and C2 (see Figure 2).
If we consider an NFV environment, where SFs are instantiated on demand as VNF instances, then the SFC Controller’s C3 interfaces to SFs map to data plane links, which are situationally used for control plane communication. Such DP configuration is by no means different from an SFP.

However, from SDN forwarding control point of view, traffic classification or forwarding is again situational. Classification may encapsulate (label) flows based on given match criteria with respect to different protocol headers, while forwarding may simply based on matching on the SFC encapsulation labels. Therefore, in a virtual data plane component SFC classification and SFF may be merged. Note, if necessary, port based capability reporting may constrain logical use of ports for classification or pure forwarding. Ports incapable of any match rule processing revert to port based forwarding.

Figure 3 shows a virtual data plane where Classifications and SFF of Figure 2 are combined into a single DP node abstraction.

Figure 2: SFC Control Plane with Virtual Data Plane Elements
Figure 3: SFC Control Plane: Overview

But there must exist a control component who is responsible for instantiating the SFs (VNFs).
We combined the SFC Control Plane with the NFV Orchestrator and the Virtualized Infrastructure Manager (VIM) into a Joint NFV and SFC Control API. The combined control (VNF and SFC) together with the virtualized data plane creates a single Big Switch with Big Software (BiS-BiS) data and control plane abstraction (see Figure 5.)
Any technology/administration/ownership domains may consist of an arbitrary topology of BiS-BiS virtualized data and control plane nodes. There exists an NFV and SFC data and control plane abstraction over which control entities can be recursively connected into a hierarchy [I-D.unify-nfvrg-recursive-programming]. An example control plane structure hierarchy is shown in Figure 6, where the "U" denotes the recurring joint abstraction control interface.
Figure 6: Recurring NFV and SFC Control Plane

5. Experimental Network Scenario

Figure 7 shows our network scenario with two layers of SFC Controllers corresponding to the technology domains and the overarching SFC Controller.

The Mininet SDN and Click Execution Environment (EE) domain contains four OVS switches with two Click EEs attached to two of the four OVS switches. A host emulation is also attached to one of the OVS switches. There is an inter-domain link connecting one of the OVS switch to the Transport SDN Domain.

The transport SDN domain consists of two MikroTik RB2011Ui HW OpenFlow switches and is controller by a POX controller integrated within the overarching SFC Controller.
The OpenStack domain is a vanilla OpenStack release with an OVS switch converting L2 steering to VxLAN connections to the Virtual Machines (VMs) running in the compute nodes.

The Docker host is extended with SFF forwarding, i.e., it natively encapsulates/decapsulates frames received over its data plane connection to and from the hosted containers.

In the example SFC request, a L3 Host is connected to a L3 WebServer in the forward direction and in the backward path a deep packet inspection (DPI), a sniffer, a header compressor (HC) and a header de-compressor (HDC) are inserted into the SFC as bump-in-the-wire (middlebox) SFs. (See Figure 7).

The SFC request to the infrastructure mapping is dynamic and is based on i) compute, storage and networking resource availabilities and capabilities; ii) constraints contained within the SFC request (e.g., latency, bandwidth, memory, etc. requirements) and iii) operational policies (e.g., utilization, energy efficiency, etc.)

The SFC example mapping shown in Figure 7 are: WebServer and DPI -> OpenStack; sniffer -> Docker; HC and HDC -> Mininet. The example is built incrementally by extending the Host -- WebServer connection by an additional SF in each step.
5.1. SFC Control Interfaces

5.1.1. C1/C2 Interfaces

Since C1/C2 interfaces are combined in our design, it is situational, which one is in use. For example, the classification and the SFP encapsulation of the Host’s traffic into the SFC is requested for the Host’s attachment point by Controller G to Controller M. Controller M in turn, determines where such classification can happen and maps the request to a network element, where the classification can be executed. For example, if a complex classification is not supported at the attachment point, then a port based forwarding is configured to convey all the Host’s traffic to a classification port. In our case, the classification is executed at the Host’s attachment port and an SFP encapsulation is assigned.
The SFP encapsulation, however, is split according to the Controllers’ responsibility domains. While Controller G is responsible the encapsulation for External Network Network Interfaces (ENNI) between the different domains, domain Controllers M, D, O are responsible for the SFP mapping between their ingress an egress points.

The ENNI encapsulations are technology specific as alignment between multiple domains must be achieved. Such domain specific parameters (e.g., supported transport layer encapsulations) are part of the domain virtualizations shown to the higher layer control entity.

### 5.1.2. C3 Interface

We use the C3 interface to configure SFs, which must be SFF aware. The VMs within the OpenStack domain must terminate/initiate L2 in L3 tunnels, in our case VxLAN tunnels.

In the case when SFs are instantiated on demand (part of an NFV orchestration together with SFC configuration), then the SF to SFC Controller connection must also be established on-demand. This, however, - as one can observer - is no way different compared to SFP connecting two or more SFs; one situationally being the SFC Controller itself. Therefore, the control or data plane usage of an SFC is situational; in a fully dynamic environment they boil down to the same configuration mechanisms.

Interestingly, conveying SFF forwarding configuration to SFs may happen via various platform services. In our case, when we only used the SF’s C3 interface to configure the VxLAN endpoints, we used OpenStack to VM metadata communication services with an agent in the VM pulling VxLAN configuration data.

However, if a direct C3 control interface is needed between the SF and the SFC Controller then a corresponding SFC must be established.

### 6. Gap Analysis

#### 6.1. SFC Requirements

##### 6.1.1. General requirements

We assess the requirements of [I-D.ietf-sfc-control-plane] one by one:

"Some deployments require that forwarding within an SFC-enabled domain must be allowed even if no control protocols are enabled. Static configuration must be allowed."
No specific considerations; both the SFs can be Physical Network Functions (PNFs) or pre-instantiated VNFs; the forwarding configuration may be statically configured.

"A permanent association between an SFC data plane element with a Control Element must not be required; (…)"

No special considerations; both SFs and the forwarding overlay works according to their configurations if connection to the control entity is lost.

6.1.2. SFC Control Plane Bootstrapping

"The interface that is used to feed the SFC control plane with service objectives and guidelines is not part of the SFC control plane itself. (…)"

"Locators for classifiers/SFF/SFs/SFC proxies, etc."

In the proposed domain abstraction only classifiers/SFF/SFC proxies are visible for the SFC Controller (the rest is not of the concern of the SFC Control, hence are hidden); therefore, the logical locators for the classifiers/SFF/SFC proxies are inherently known

SFs are instantiated by the NFVO logic co-existing with the SFC controller. Therefore, the combined Control Plane inherently knows the location of all SFs.

"SFs serviced by each SFF"

This is the control plane association; hence inherently exists.

"A list of service function chains, including how they are structured and unambiguously identified."

The structure of SFC is created within the SFC Control Plane, hence it inherently exists there.

Status of each SFC: active/pre-deployment phase/etc.(…)"

In the experimental implementation these states are internal to the Control Plane and they do not exist below the SFC Controller. This is because an SFC Controller configures the underlying virtual data plane instead of communicating with SFC intents. However, there may exist various configuration targets in the could be running, candidate, initial, etc. or other configuration
targets may exist, however, those are independent generic configuration targets.

"A list of classification guidelines and/or rules to bind flows to SFCs/SFPs."

- In our view, all these parameters are input to the SFC Controller and are part of the SFC request. As such, the goal of the SFC Controller is to map such northbound requests to its southbound interfaces. All the requests, the mapping and the southbound outputs are inherently stored.

"Optionally, (traffic/CPU/memory) load balancing objectives at the SFC level or on a per node (e.g., per-SF/SFF/SFP proxy) basis."

- In our design, the northbound to southbound mapping algorithm takes into account these objectives as part of the operational policies or explicit tenant requirements contained in the SFC request. It is important to note, that such requirements are service agnostic.

"Security credentials."

- In our design, each tenant has its own SFC Control plane interface with its full context (policy, access, accounting, security, etc.). Our SFC API is build over the NETCONF protocol [RFC6241] relying on a secure transport layer.

"Context information that needs to be shared on a per SFC basis."

- We maintain per tenant context; individual SFCs are situational with this respect as a tenant may merge/split SFCs any time. The SFC Controller builds configuration request for the underlying layers aggregating all of its tenants’ request, therefore SFCs are always scoped to control API and does not travel individually through the vertical SFC control plane layers.

7. Open Questions

- What is the rationale behind the separation of the SF placement and the SFC Control Plane (traffic steering) if the SFC Control Plane needs to know the locators of the SF? In a dynamic environment, when SFs are deployed as VNFs, can a joint consideration of software and networking resources result in better deployments?

- What is the rationale behind the coupling of the Service Path Header with the Context Headers? Can the Service Path Header
stripped from the encapsulation before sending it the SF? Is the Context Header service specific? If separated, can the Service Path Header be encoded into the forwarding behavior of the domain and yield all SFs SFP (overlay) unaware?

8. IANA Considerations

This memo includes no request to IANA.

9. Security Considerations

TBD

10. Acknowledgement

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 619609 - the UNIFY project. The views expressed here are those of the authors only. The European Commission is not liable for any use that may be made of the information in this document.

11. Informative References

[I-D.ietf-sfc-control-plane]

[I-D.unify-nfvrg-challenges]

[I-D.unify-nfvrg-recursive-programming]
Szabo, R., Qiang, Z., and M. Kind, "Towards recursive virtualization and programming for network and cloud resources", draft-unify-nfvrg-recursive-programming-02 (work in progress), October 2015.


[UNIFY-GLOBECOM]

Authors’ Addresses

Robert Szabo
Ericsson Research, Hungary
Irinyi Jozsef u. 4-20
Budapest 1117
Hungary

Email: robert.szabo@ericsson.com
URI: http://www.ericsson.com/

Balazs Sonkoly
Budapest University of Technology and Economics
Magyar tudosok krt. 2
Budapest 1111
Hungary

Email: sonkoly@tmit.bme.hu
URI: http://www.tmit.bme.hu/