This is the final peer-reviewed accepted manuscript of:

G. Davoli, W. Cerroni, C. Contoli, F. Foresta and F. Callegati, "Implementation of service function chaining control plane through OpenFlow," ***2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*****, Berlin, 2017, pp. 1-4.**

The final published version is available online at DOI:

https://doi.org/10.1109/NFV-SDN.2017.8169852

Rights / License:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/)

When citing, please refer to the published version.

Implementation of Service Function Chaining Control Plane through OpenFlow

Gianluca Davoli, Walter Cerroni, Chiara Contoli, Francesco Foresta, Franco Callegati
DEI - University of Bologna, Italy
Email: {gianluca.davoli, walter.cerroni, chiara.contoli, francesco.foresta, franco.callegati}@unibo.it

***Abstract*—This papers describes a proof-of-concept implementation of the Service Function Chaining Control Plane, exploiting the IETF Network Service Header approach. The proposed implementation combines the OpenFlow protocol to control and configure the network nodes and the NSH method to adapt the service requirements to the transport technology. The manuscript shows that the result of this combination is a very general architecture that may be used to implement any sort of Service Function Chain with great flexibility.**

I. Introduction

With the recent widespread adoption of virtualization technologies and network overlays, many network components, including forwarding devices, servers and applications, have undergone significant changes. Nevertheless, the way services are deployed into a network remained substantially unchanged, impacting the speed at which critical applications can be deployed, and significantly increasing operational costs for network operators.

The expression *Service Function Chaining* (SFC) is generally used to describe the deployment of composite services that are obtained from a concatenation, i.e., a *chain*, of one or more basic services. In other words, a SFC[1] is fundamentally the series of service functions that a packet or flow must traverse. Deploying a SFC in traditional infrastructures requires significant configuration and management efforts on vendor-specific appliances. As a consequence, Virtualized Network Functions (VNFs) and the Network Function Virtualization (NFV) paradigm [1] are attracting the interest of operators because they promise a decoupling of the logical functionalities from the underlying hardware, with potential significant reductions in CAPEX and OPEX. At the same time, the deployment of SFCs can make use of Software Defined Networking (SDN) principles for efficient and flexible control and management purposes [2].

SFC makes use of a service-specific overlay that creates the required service topology. Therefore SFC inherently defines a Service Plane, that is an intermediate plane between Application and Control Planes. The Service Plane includes all the processes that allow the infrastructure to provide services to users and maintains state on those services, relying on Control and Management Plane functions to suitably program the Data Plane.

[1]In this manuscript the SFC acronym will be used to refer to both Service Function *Chaining* and Service Function *Chain*, depending on the context.

Several aspects of SFC are currently being investigated by the research community. SFC Orchestrators to deploy SFCs as well as control their activity and make adjustments are introduced in [3]. The problem of allocating physical resources to data plane components of a SFC is addressed in [4], while a solution for the trade-off between optimized performances and resource cost in SFC deployments is presented in [5].

A very important problem in the implementation of the SFC Orchestrator arises when the chain spans several network domains with non homogeneous forwarding technologies. This problem was addressed by the Internet Engineering Task Force (IETF) in [6], where it is suggested that the service-specific overlay can be obtained by applying packet encapsulation. One option being considered by IETF is the so-called *Network Service Header* (NSH) [7], which intends to provide a flexible, dynamic, and transport-independent SFC solution for the data plane. The NSH draft focuses on data plane aspects only, and very little has been said about a possible SFC control plane solution. To the best of our knowledge, the only document that attempts to do so is another IETF draft that, at the time of writing, has already expired [8].

In this paper we propose a possible implementation of a NSH-aware control plane inspired by the concepts discussed in [8]. Our approach is based on the use of SDN-like technology inside NSH nodes and on the adoption of the OpenFlow protocol for the communication between the SFC Control Plane and the Service Plane components. We present here the main idea and a preliminary proof of concept.

The rest of the paper is structured as follows. In Section II we briefly recall the SFC Architecture. Then, in Section III we propose our solution for the implementation of SFC Control Plane functionality. In Section IV we show some experimental validation of our proposal, before presenting our conclusions in Section V.

II. Service Function Chaining Architecture

The SFC architecture introduces some important concepts that we briefly mention here [6]. The *Service Function Path* (SFP) is a specification of the path to be followed by packets assigned to a certain SFC. It is an abstraction of the sequence of nodes the packets requiring a given service will traverse. On the other hand, the *SFC encapsulation* (SFC-En) always provides SFP identification and can optionally provide further information. It is used by the SFC-aware functions to realize the Service Plane functionalities, but it is not used for packet

forwarding through the underlying network topology. Carrying the SFC-encapsulated traffic is the task of the chosen network transport protocol.

The main components of the SFC Service Plane are:

- *SFC Classifiers* (SFC-Cl), which classify the incoming traffic based on predefined policies, in order for the flow to be steered through the required set of network service functions; the main task for the SFC-Cl is to add the SFC-En, which is then removed by the last node in the SFP, or by a SFC-aware function that consumes the packet;
- *Service Functions* (SF), which are the basic elements of a chain, and are responsible for a specific treatment of received packets; they can act at different levels of the protocol stack, and they can be implemented either as virtual elements hosted by a server, or as physical equipment with specialized hardware; a SF can be either SFC-aware (i.e., able to act on SFC-encapsulated packets) or SFC-unaware (i.e., it must receive only packets without SFC encapsulation);
- *Service Function Forwarders* (SFF), which are responsible for forwarding traffic to one or more connected SFs according to information carried in the SFC-En; they can also terminate the SFP;
- *SFC Proxies* (SFC-Pr), which remove and insert SFC-En on behalf of SFC-unaware SFs, before and after their action, respectively.

The reference architecture of the SFC Control Plane (SFC-CP) described in [8] defines the following interfaces to communicate with Data Plane components:

- interface *C1*, between SFC-CP and SFC-Cl, used to manage SFC classification rules in classifiers;
- interface *C2*, between SFC-CP and SFF, used for exchanging required information for SFC forwarding decision-making, collect state information on SFPs, etc.;
- interface *C3*, between SFC-CP and SFC-aware SF, used, for example, to collect output information resulting from the processing of packets in the SF;
- interface *C4*, between SFC-CP and SFC Proxies, used to communicate SFC instructions and to retrieve state information.

The deployment of SFCs must take into account complex aspects that must be handled carefully, as reported in [9]. Such aspects include topological dependence, consistent ordering of SFs, and dynamic SFC classification. Moreover, end-to-end SFCs are typically deployed across multiple network administrative and/or geographical domains.

The SFC Architecture can be implemented by making use of NSH, which defines a Service Plane protocol, specific for the creation of dynamic SFCs. It provides SFP identification, transport-independent chaining, and packet-based network and service metadata. NSH is designed to be easy to implement across a range of devices, both physical and virtual, including hardware platforms.

The two most important fields in the NSH header are the *Service Path Identifier* (SPI) and the *Service Index* (SI). The SPI is a 24-bit integer number assigned to packets by the first SFC-Cl in the SFP, and all nodes taking part in that SFP must use the same SPI consistently. The SI, an 8-bit integer number, is used to identify the location within the SFP. The SI must be set by the initial SFC-Cl either to its maximum value (i.e., 255) or to a value related to the length of the SFP, and it must be decremented by one unit by all SFC-aware SFs and SFC Proxies the packet traverses in the SFP.

III. OpenFlow-based NSH Control Plane

The reference scenario for the proposed NSH control plane is shown in Fig. 1. It is composed of a SFC-CP entity, a pair of SFC-Cls, an intermediate node serving as both SFF and SFC-Pr towards SFC-unaware SFs, a SFC-aware SF, and two SFC-unaware SFs. In our reference implementation we assume that each Service Plane entity is built around an OpenFlow-capable switch (OF-S). Then, all SFC entities are interconnected by means of a tunneling technology (e.g., VXLAN) through an underlying network infrastructure, controlled by one or multiple network operators through a generic control plane paradigm. The network infrastructure can use either SDN or non-SDN control, but this does not matter because the proposed SFC-CP is separate from the network control plane. Therefore, *service providers and network providers can act as completely independent entities, each adopting its favorite control plane approach*.

Mapping a SFP to the transport network requires to define a relationship between a given position in the SFP (i.e., a SPI/SI pair) and a certain next-hop in the underlying network. While the former information belongs to the Service Plane, the latter depends on the network's topology and technology, as it must point to an existing location in the underlying network, typically expressed as an address (e.g., IP or MAC). How to implement this mapping is not a matter of standardization and different solutions may be adopted. This paper introduces a mapping strategy that, to the best of our knowledge, has not been proposed yet. It is a rather straightforward idea: *mapping the SFP-to-transport relationship onto the ports of the employed OF-S*.

In our implementation, each NSH interface, corresponding to a specific SPI/SI pair, is bridged to a port on the node's internal OF-S. The working principle of the proposed implementation is the following: through the association of SPI/SI pairs to ports on a OF-S, it is possible to have the node acting as a NSH Service Plane component while controlling it through the OpenFlow protocol from an SDN Controller, which takes the role of SFC Control Plane entity (SFC-Co) running applications that enforce Service Plane policies.

The NSH mapping tables are therefore implemented in the form of flow tables inside the OF-S. As an example, assume port N of the OF-S is bridged to interface `nshM` of the node. Instructing the switch to send traffic out of port N will result in the node sending NSH-encapsulated traffic out of interface `nshM` with the corresponding SPI/SI values. Therefore, depending on what kind of flow rules are installed in the internal OF-S, a SFC node can be programmed to perform different

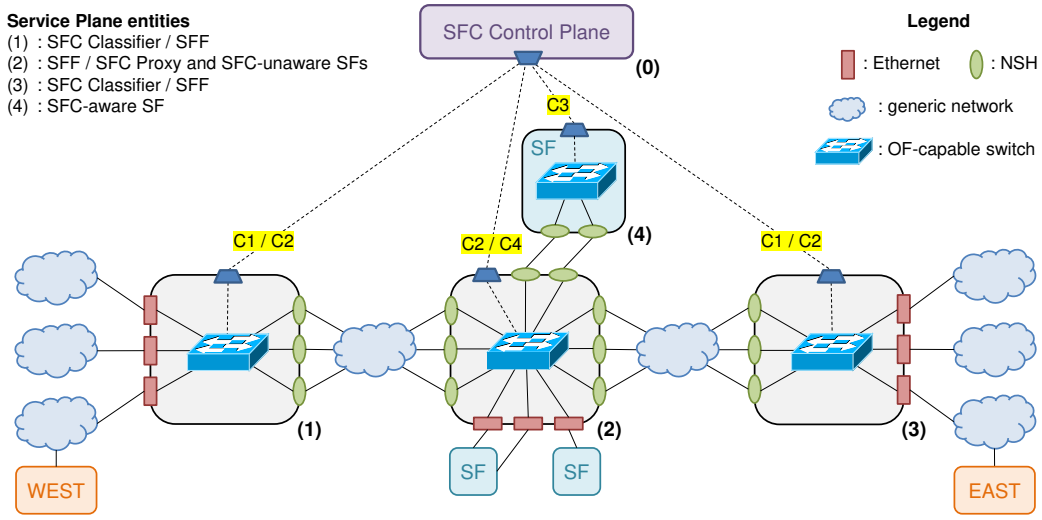

Fig. 1. Reference scenario: the role of Nodes (1) to (4) is shown in the upper left corner.

Service Plane entity functions. With reference to Fig. 1, the entities are mapped to the nodes in the following way:

- Node (0) hosts the SFC-Co.
- Node (1) is responsible for adding the NSH tag to packets coming from *WEST* hosts and forwarding NSH-encapsulated packets to the first SFF in the SFP: in this role, it acts as SFC-Cl. Additionally, this node is also responsible for removing the NSH tag from packets assigned to a SFP which ends at Node (1), such as packets destined to *WEST* hosts, thus acting as SFF. Following this approach, the SFC classification is as expressive as OpenFlow matching is.
- Node (2) is responsible for handling the NSH encapsulation on behalf of SFC-unaware SFs, as well as for forwarding the NSH-encapsulated packets to the following SF or SFF in the SFP. In those two tasks, Node (2) acts as SFC-Pr and SFF, respectively.
- Node (3), similarly to Node 1, acts both as SFC-Cl and SFF for the traffic exchanged with *EAST* hosts.
- Node (4) acts as a SFC-aware SF, as it is able to receive NSH-encapsulated packets from the SFF and process them, before sending them back to the SFF after updating the SI.

IV. Experimental validation

A. Test bed setup

As a proof of concept, we developed a test bed to implement the proposed solution, based on the reference scenario illustrated in Fig. 1. The test bed comprises a total of five Virtual Machines (VMs) and the interconnecting virtual networks. The VMs are deployed on a single physical server, and virtualization is managed through libvirt/KVM. All of the involved VMs run Ubuntu 14.04 LTS. One of them hosts an instance of the ONOS SDN Controller [10], while the remaining four VMs implement NSH-capable nodes. The choice of ONOS as SDN Controller (therefore, in this test bed, as SFC-Co) is motivated by its availability of Java and REST APIs, as well as of a well-documented Command Line Interface (CLI) and Graphical User Interface (GUI), which allow for easier monitoring of the controller's activities. However, this choice does not affect the generality of the implementation.

On each NSH node we installed and enabled the open-source NSH kernel module [11]. We assigned a SPI/SI pair to each NSH interface, and mapped each of them to a transport-level next-hop (i.e., an IP address), instructing the node to use VXLAN as encapsulation protocol to obtain the overlay topology. This is equivalent to adding an entry in the NSH-to-transport mapping table specifying that all traffic addressed to the endpoint with that SPI/SI should be encapsulated in VXLAN packets and sent to the specified remote IP address. Similarly each NSH interface was made aware of the inbound SPI/SI values it is meant to receive. Thus the mapping was achieved for outgoing and incoming traffic.

The transport network infrastructure will be traversed by as many VXLAN tunnels as the number of SPI/SI pairs defined. Each packet sent out by the VMs over one of their NSH interfaces will be intercepted by the NSH kernel module and encapsulated in a NSH/VXLAN packet, obtaining the SPI/SI pair assigned to the NSH interface. Similarly, when a packet is received on one of the NSH interfaces, the kernel module will intercept it and remove the NSH/VXLAN encapsulation, before handling the packet to the traditional IP forwarding module of the VM.

As outlined above the NSH nodes were created exploiting the Open vSwitch bridge (OvS) as internal OF-S, programmed by the SFC-Co. We attached the previously defined NSH logical interfaces to the OvS ports. The *WEST* and *EAST* hosts as well as the SFC-unaware SFs were implemented as logically isolated virtual entities by means of Linux network namespace technology. We implemented the full set-up depicted in Figure 1. As SFC-unaware SFs, we deployed a *Deep Packet Inspector* (DPI) and a *Traffic Controller/Shaper* (TC),

the latter configured with two Layer-2 interfaces (inbound and outbound). The SFC-aware SF is an *Integrity Checker* (IC). The *WEST* hosts represent users wishing to communicate with the *EAST* hosts with different priorities and the following service policies:

- traffic coming from user WEST1 should be first checked by the DPI and then copied in the IC;
- traffic coming from user WEST2 should be first checked by the DPI and then limited in bandwidth by TC.

Therefore, three possible SFCs are needed with their respective SFPs:

- SFC1, from any WEST user to the destination EAST user, duplicating the traffic towards the DPI;
- SFC2, from a high-priority WEST user to the destination EAST user, passing through the IC;
- SFC3, from a low-priority WEST user to the destination EAST user, passing through TC for bandwidth limitation.

B. Proof-of-Concept validation

We deployed a basic orchestrator (implemented as a script emulating an orchestrator's interaction with ONOS) in the SFC-Co node, which accomplishes the desired dynamic SFC behavior. The orchestrator installs proactive flow rules in the OF-S internal to relevant SFC entities, so as to apply chain SFC1. Then, it waits for any WEST user to start a flow of traffic towards the destination EAST user. When the flow starts, the orchestrator starts the DPI, and after a small time period, it retrieves information from it. If the inspected traffic contained data from WEST1, the script installs rules applying SFC2, otherwise, if the traffic contained data from WEST2, the script installs rules applying SFC3. It should be noted that traffic flows are steered to a different SFP without stopping them, thus achieving dynamic SFC.

The WEST-to-EAST throughput measured at the OF-S within Node (2) while applying the different SFCs is shown in Fig. 2. At first, SFC1 is applied to traffic from WEST1 (from $t = 8$s to $t = 20$s), then after inspection SFC2 is applied (from $t = 21$s to $t = 38$s). Later on, traffic from WEST2 is subject to SFC1 (from $t = 53$s to $t = 65$s), then after inspection SFC3 with shaping is applied (from $t = 66$s to $t = 84$s). This outcome proves the correct implementation of dynamic SFC in our test bed.

V. Conclusion

The SFC Control Plane solution proposed in this paper is based on the SDN paradigm. In particular, assuming SFC entities that are built around an OpenFlow-capable switch we can take advantage of SDN's inherent dynamicity and programmability also in the Service Plane, while keeping it independent of the underlying network infrastructure. An interesting by-product is that network providers and service providers can adopt completely separate Control Plane solutions. We validated our approach on a test bed emulating multiple SFC entities interconnected by non-SDN networks under dynamic chaining scenarios.

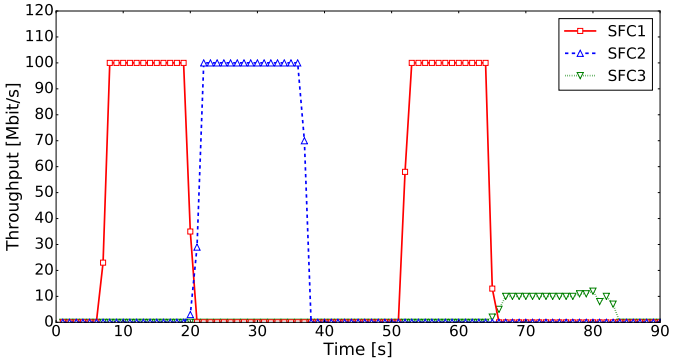

Fig. 2. WEST-to-EAST throughput measured at the OF-S within Node (2) while applying dynamic SFC.

Acknowledgment

The authors would like to thank Ms. Chiara Di Nenno, for her contribution in the development of a shared knowledge on the NSH Control Plane. This work has been partially supported by project "GAUChO - A Green Adaptive Fog Computing and Networking Architecture," funded by the Italian Ministry of Education, University and Research (MIUR) under the "PRIN Bando 2015" program, grant no. 2015YPXH4W_004.

References

[1] "Network Functions Virtualisation (NFV); Architectural Framework," The European Telecommunications Standards Institute (ETSI), October 2013. [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/nfv

[2] F. Callegati, W. Cerroni, C. Contoli, R. Cardone, M. Nocentini, and A. Manzalini, "SDN for dynamic NFV deployment," *IEEE Communications Magazine*, vol. 54, no. 10, pp. 89–95, October 2016.

[3] A. M. Medhat, G. A. Carella, M. Pauls, M. Monachesi, M. Corici, and T. Magedanz, "Resilient orchestration of Service Functions Chains in a NFV environment," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2016, pp. 7–12.

[4] M. T. Beck, J. F. Botero, and K. Samelin, "Resilient allocation of Service Function Chains," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2016, pp. 128–133.

[5] T. Soenen, S. Sahhaf, W. Tavernier, P. Skldstrm, D. Colle, and M. Pickavet, "A model to select the right infrastructure abstraction for Service Function Chaining," in *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Nov 2016, pp. 233–239.

[6] J. M. Halpern and C. Pignataro, "Service Function Chaining (SFC) Architecture," RFC 7665, Oct. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7665.txt

[7] P. Quinn and U. Elzur, "Network Service Header," Internet Engineering Task Force, Internet-Draft draft-ietf-sfc-nsh-12, Feb. 2017, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-sfc-nsh-12

[8] M. Boucadair, "Service Function Chaining (SFC) Control Plane Components," Internet Engineering Task Force, Internet-Draft draft-ietf-sfc-control-plane-08, 2016, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-sfc-control-plane-08

[9] T. Nadeau and P. Quinn, "Problem Statement for Service Function Chaining," RFC 7498, Apr. 2015. [Online]. Available: https://rfc-editor.org/rfc/rfc7498.txt

[10] ONOS: Open Network Operating System. [Online]. Available: http://onosproject.org

[11] Network Service Header Linux kernel module implementation. [Online]. Available: https://github.com/upa/nshkmod