Service Function Chaining: a lightweight container-based management and orchestration plane

Armir Bujari
and Claudio E. Palazzi
Department of Mathematics
University of Padua
Padua, Italy
Email: {abujari,cpalazzi}@math.unipd.it

Davide Polonio
and Marco Zanella
Department of Mathematics
University of Padua
Padua, Italy
Email: {davide.polonio,marco.zanella.8}@studenti.unipd.it

***Abstract*—The increase in the amount of traffic with heterogeneous QoS requirements poses several challenges to end-to-end service provisioning. Network resource management and service differentiation are two crucial functionalities employed by service providers, proven essential in order to meet the end user requirements. In particular, application-driven networking has emerged as a viable alternative embodying the potential of enabling specific flow optimizations meeting application-specific requirements. In this context, Service Function Chaining (SFC) is seen as an enabling technology for the flexible management of specific service/application traffic. In this paradigm the network functions are seen as an ordered chain of interconnected functions handling traffic delivery (data plane), control and management. In this article we present a work in progress of a virtualized, container-based testbed with orchestration and management capabilities supporting service chains.**

***Index Terms*—NFV, SDN, 5G, Service Function Chaining, Management and Orchestration**

I. Introduction

Many hopes have been put into the future 5G infrastructures with the IMT-2020 specification [1] qualifying for speeds of gigabits down/up-link and with up to a couple of milliseconds end-to-end latency. However, commercial deployments of 5G networks are not expected before 2022 while the Internet traffic is steadily experiencing an exponential growth and it could quickly catch up (as the trend in multimedia content consumption shows) exceeding this capacity baseline [2]. From this basis, it is evident that we need to make peace with the technology at hand and optimize for resource utilization in current networks. This task is often too cumbersome when considering the lack of flexibility of the network elements [3].

In this context, Network Function Virtualization (NFV) is an emergent paradigm proposed in the telecommunications domain whereby network functions are decoupled from the hardware and can run in software anywhere in the network [4]. Network elements are no longer seen as a collection of integrated hardware and software entities but can evolve and be deployed independently providing a much needed flexibility to scale the actual performance in a more dynamic way. To achieve this objective the NFV architectural framework introduces the following elements (depicted in Fig. 1):

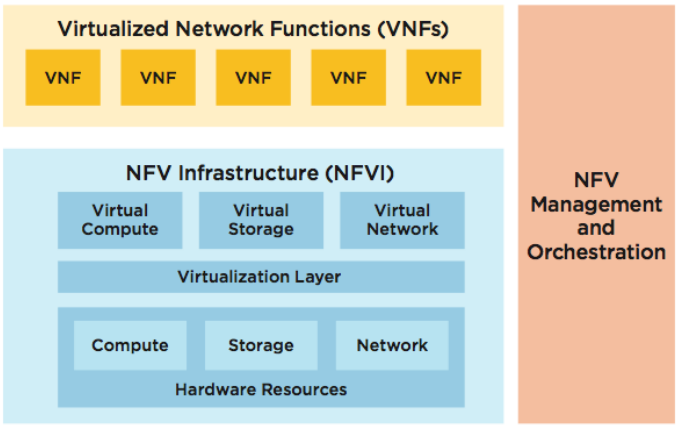

Fig. 1. Network Function Virtualization reference architecture.

1) Virtualised Network Function, as the software implementation of a network function which is capable of running over the NFVI;
2) NFV Infrastructure (NFVI) includes the diversity of physical resources (storage, compute and networking) and how these can be virtualised, deployed to meet the needs of VNFs;
3) NFV Management and Orchestration, which covers the orchestration and lifecycle management of physical and/or software resources that support the infrastructure virtualisation, and the lifecycle management of VNFs.

The virtualization layer is actually the hypervisor, which is responsible for abstracting the hardware resources of a compute domain. One might, e.g., have a single physical server (physical memory and physical compute) but the hypervisor can partition it into multiple virtual memories and virtual computes in such a way as to make each entity independent from all of the others. That is why the virtualization layer, together with the virtual resources, is also referred to as the Hypervisor domain. For a thorough discussion on the NFV architectural components, we refer the reader to the technical specification [4].

Our proposal somehow departs from the classical interpre-

tation based on the hypervisor-oriented approach. Indeed, we propose to replace virtual machines with containers, which allow for a more lightweight approach to virtualization, by removing the need for the guest operating system. We refer, in particular, to the so-called Kubernetes tool that has been conceived at the outset as a means for defining and running multi-container applications starting from a file-based description of the target infrastructure [5]. Such an approach is inline with the latest proposals associated with the so-called Infrastructure as Code (IaC) paradigm, which has proved to be extremely useful when applying continuous integration and testing principles.

The rest of the article is organized as follows: In Sec. II we provide some background information related to on-going standardization effort of the service function chaining paradigm. Next, in Sec. III we provide a synthetic overview of prior work, providing SFC support in the management and orchestration plane. Section IV discusses a two-layer service orchestration framework, highlighting some features and software components. Finally, Sec. V draws the conclusions and presents on-going future work.

II. Background

The SFC network model, as specified by the IETF SFC WG [6], is comprised of two main layers, namely the control and data plane. The control plane provides the management and orchestration functionalities of the chains and their logical paths and is responsible for deploying the policies into the SFC Classifier, operating at different levels of the protocol stack, differentiating the traffic and assigning it to the corresponding Service Function Path (SFP). The service chain is seen as a logical order of SFs on a network path, defined by the operator using control plane mechanisms. A SF is a network function that performs specific functionality to the traffic then returns it back to the Service Function Forwarder (SFF) such as DPI, NAT, etc.

The data plane involves the SFs where the traffic goes through as imposed by control plane logic. It is composed of an ordered set of SF instances tied together by a single SFF or multiple SFFs. This entity is responsible for forwarding the traffic to SFs or other SFFs according to the information carried in the packet based on the assigned SFP. The SFC classifier is a logical classifier containing policies and applying them into the ingress SFF in order to classify the traffic based on the target application or service (video streaming, voice call, download etc) or other predefined policies.

In Fig. 2 we show an exemplification of the SFC network model, where SFC1 and SFC2 refer to different chains applicable to two different provisioned applications, respectively.

Clearly these separation of duties fits well within the Software Defined Network (SDN) paradigm advocating for a separation of control and data plane, providing the necessary primitives for a reconfigurable data layer, achieved through well-defined interactions through the so-called southbound (data-control) interface.

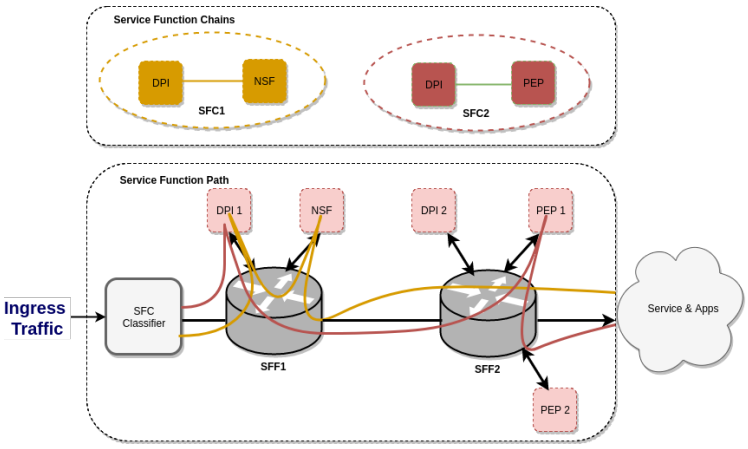

Fig. 2. Exemplification of the Service Function Chaining approach.

An add-on in this depicted frame is brought by the NFV architectural framework advocating for a standardized Management and Orchestration (MANO) plane, porting cloud-computing related concepts into the picture. The management and orchestration plane further decouples some of the responsibilities of the control-plane, introducing into the picture a centralized brain with end-to-end visibility of network resources, being those compute, network or storage. These interactions occur through standardized references points and different integration patterns of SDN/NFV are envisaged depending on the context [7]. Also, the NFV specification defines Virtual Network Function Forwarding Graph (VNFFG), which is the equivalent of SFC from the ETSI perspective.

III. Related Work

Several proposals implementing SFC relying on SDN and/or NFV frameworks can be found in literature [8]. Furthermore, different open source implementations of SFC ETSI-compliant Orchestration are available providing some capabilities for SFC orchestration. These proposals differentiate depending on the SFC orchestration capabilities and the actual approach employed to provision it.

A first criteria differentiating the proposals is whether the SFC implementation adopts either NFV or SDN or both technologies in a unified architecture. Where no orchestration plane is available, this feature is implemented as an SDN application in tight interaction with the SDN control plane through the north-bound interface. When both SDN and NFV are employed different integration patterns have been employed depending on the deployment context.

Also, the proposals differ in the actual implementation of the SFC forwarding plane. The SFC standard proposes a dedicated header, an above-network layer encapsulation header, the Network Service Header (NSH), containing information used to forwarding packets in the service chain which can be further enriched with context information depending on the deployment scenario. Many solutions do not rely on NSH to enact the forwarding process, rather employ layer 2 or 3

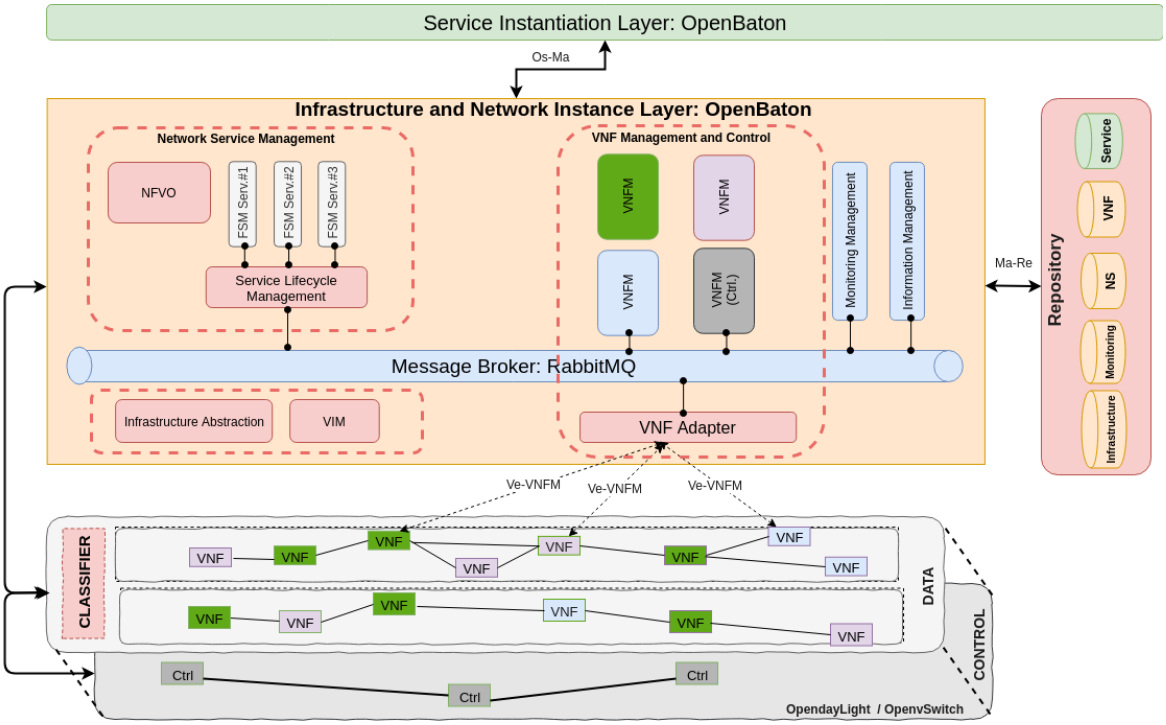

Fig. 3. Software architecture, main functional components.

information. This is partially as a consequence to the lack of support of NSH encapsulation in SDN products.

Another important aspect of SFC, is the capability of the dynamic provisioning of service chains. This is achieved through the availability of monitoring and management functions capable of instantiating and scaling the network functions (resources) accordingly. Traffic steering or engineering is of paramount importance and this capability is achieved through a tight interaction of control-data-management plane.

As an example, proposals like [9], [10] provision static service chains and limited capabilities of traffic engineering while still showing the benefits of the approach with respect to the traditional middlebox deployment. To the best of our knowledge, current approaches rely on a VM-based orchestration model, achieved through Openstack [11], lacking the integration and support for lightweight, container-based technologies. Filling this gap, we adopt a container-based approach to SFC, integrating Kubernets in an ETSI-compliant MANO. While its current status, the proposal instruments only container-based technology, it is not limited to it.

IV. Proposal

The softwarization of the network and inter-networking functions is key to meeting the objectives of our proposal, providing the much-needed flexibility to orchestrate, manage and deploy, on a per need basis, the network functions on the end-to-end delivery path. The proposed approach aims at researching and experimenting with the synergistic combination of the container-based and VM-based approaches.

A. Software Architecture

The envisioned software architecture of our approach is shown in Fig. 3, providing a high-level description of the components in a unified frame. The proposal envisions a repository of network resources (either software, as virtual functions, or hardware) providing the necessary abstractions for the management and monitoring plane to express intents over infrastructure through policy.

The ETSI NFV architectural framework is service/application agnostic and in particular is does not deal with:

1) Application-aware NS configuration and management.
2) VNF application layer configuration and management.
3) Management and deployment of PNFs, or their application layer configuration and management.

To this end, a service-aware software layer is required, tasked with service lifecycle management and monitoring through a well-defined set of established APIs with the ETSI MANO. Each orchestrator, hereafter informally called higher layer (service) and lower layer orchestrator, has its own duties and responsibilities in provisioning the end-to-end services. Without loss of generality, both orchestration layers rely on OpenBaton NFVO, chosen for its ease of use and capability to be deployed as a containerized software instance [12]. The depicted functional components exchange messages through the RabbitMQ message middleware, an open source message broker implementing the Advanced Message Queuing Protocol. The level of distribution of the components is a matter

of configuration as the messaging middleware abstracts the software components form these technicalities.

Zooming into the lower layer orchestrator, its job and functionalities are those belonging to an ETSI MANO, that is, infrastructure management and orchestration, VNF and NS lifecycle management to name a few. The MANO relies on a data repository comprised of a (non exhaustive list):

1) Virtualized network function repository containing the software entities enabling the envisaged used cases.
2) Network service and network sub-slice repository containing the definition(s) of network service and available sub-slices. A Network service is comprised of a single or composite VNFs aimed at some functionality while network sub-slices or network slice subnet(s) are comprised of network, compute and storage resources, which cannot be operated in isolation as a complete network slice. These repositories are depicted together as they logically represent composite information and for ease of representation; however, in an actual deployment they might reside in different catalogues.
3) Monitoring repository containing monitoring data sourced by different components of the envisaged architecture, which can be tapped and consulted by either humans or software entities in order to, e.g., enact specific policies.

In this context, the lower layer MANO exposes a set of APIs (interfaces: Os-Ma, Os-Ma-Nfvo) to the upper layer (service) MANO, through which is possible to configure, instantiate and terminate a service. Operationally, the lower layer MANO exposes to the service layer a series of NFV resources, including but not limited to the NS repository through which it is possible to define the desired optimized service for the use-case in mind. In addition to the service definition and instantiation, the service MANO can receive NFV service-related state information.

B. A Container based SFC Approach

The proposed approach relies on a hybrid deployment model, employing both container and virtual machine based technology. In addition, a network blueprint is maintained, accounting for the resource deployment and usage status in the infrastructure domain. That is, the proposal envisions the adoption of a catalogue of network resource status (either software, as virtual functions, or hardware) providing the necessary abstractions for the management and monitoring plane to express intents over infrastructure through policy.

The system supports a feedback loop through an active monitoring, feature provided by i.e., Grafana and Prometheus, used to evaluate and drive network operations for scaling, moving and replicating network functions along the delivery chain. The dynamic (micro-)service chaining is a crucial aspect of the proposal, where depending on the monitored performance and flows adopted protocol stack, ad-hoc customization might prove more beneficial.

The architecture is based on a distributed, large scale deployment of clusters of containers, whose management and orchestration is delegated to a model-based configuration framework, namely Kubernetes. In the actual implementation we rely on this framework for the instantiation, deployment and scaling of network functions. All these functionalities are handled by Kubernetes itself, integrated with Open Baton through the messaging system.

V. Conclusion

In this article we presented an overview of NFV and SDN approaches to service function chaining, the latter a paradigm deemed of paramount importance in the next-generation networks. Rather than relying solely on VM-based orchestration, our approach integrates lightweight, container-based integration through the adoption of Kubernetes. Kubernetes is being integrated in Open Baton, providing monitoring and dynamic management capabilities of the virtualized infrastructure. While the current integration is confined to, e.g., instrumentation of container-based technology it is not limited to it. Indeed, we are currently planning an extension to the platform, allowing for a hybrid deployments model, encompassing VM-based legacy functions in the service chain. Also, work is on-going in the front of the softwarization of the control/data plane, integrating the SDN paradigm into our architecture.

Acknowledgment

This work is partially supported by funds from the ESA ARTES VIBeS project.

References

[1] M. J. Marcus, "5G and IMT for 2020 and beyond [Spectrum Policy and Regulatory Issues]," in IEEE Wireless Communications, vol. 22, no. 4, pp. 2-3, August 2015.

[2] A. Bujari, M. Massaro, C. E. Palazzi, "Vegas over access point: Making room for thin client game systems in a wireless home", in IEEE Transactions on Circuits and Systems for Video Technology, vol. 25, no. 12, pp. 2002-2012, 2015.

[3] M. Pozza, A. Rao, A. Bujari, H. Flinck, C. E. Palazzi and S. Tarkoma, "A refactoring approach for optimizing mobile networks," In Proc. of IEEE International Conference on Communications (ICC), Paris, 2017, pp. 1-6.

[4] ETSI GS NFV-IFA 010: Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Functional requirements specification. [Online]. Last Accessed: August 2018.

[5] D. Bernstein, "Containers and Cloud: From LXC to Docker to Kubernetes," in IEEE Cloud Computing, vol. 1, no. 3, pp. 81-84, 2014.

[6] C. Pignataro and J. Halpern, Service Function Chaining (SFC) Architecture, document RFC 7665, Internet Engineering Task Force, Oct. 2015.

[7] ETSI GS NFV-EVE 005: Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework. [Online]. Last Accessed: March 2018.

[8] A. M. Medhat *et. al*, "Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges," in IEEE Communications Magazine, vol. 55, no. 2, pp. 216-223, February 2017.

[9] A. Abujoda, P. Papadimitriou, "MIDAS: Middlebox Discovery and Selection for On-path Flow Processing," in Proc. of Int'l. Conf. Communication Systems and Networks (COMSNETS), pp. 1-8, 2015.

[10] J. Soares *et. al*, "Toward a Telco Cloud Environment for Service Functions," in IEEE Communications Magazine, vol. 53, no. 2, pp. 98-106, 2015.

[11] OpenStack. https://www.openstack.org. [Online]. Last Accessed: June 2018.

[12] Open Baton. https://openbaton.github.io. [Online]. Last Accessed: March 2018.