Experimenting latency-aware and reliable service chaining in Next Generation Internet testbed facility

M. Gharbaoui*, C. Contoli‡, G. Davoli‡, G. Cuffaro*, B. Martini*,
F. Paganelli*, W. Cerroni‡, P. Cappanera †, P. Castoldi§
*CNIT, Italy ; †University of Florence, Italy
‡DEI - University of Bologna, Italy ; §Scuola Superiore Sant'Anna, Pisa, Italy

***Abstract*—In this paper we report experimental validation of an orchestration system for geographically distributed Edge/NFV clouds, supporting end-to-end latency-aware and reliable network service chaining. The orchestration system includes advanced features such as dynamic virtual function selection and intent-based traffic steering control across heterogeneous SDN infrastructures. The experiment run under the 1st Open Call of the Fed4FIRE+ EU H2020 Project and took advantage of bare metal servers provided by the Fed4FIRE platform to set up a distributed SDN/NFV deployment. We provide details on how this orchestration system has been deployed on top of Fed4FIRE facility and present experimental results assessing the effectiveness of the proposed orchestration approach.**

***Index Terms*—SDN; NFV; orchestration; federated testbed; service chaining.**

I. Introduction

With the advent of Software-Defined Networking (SDN), Network Function Virtualization (NFV) and the comprehensive 5G context [1], novel service scenarios can be conceived (e.g., cloud robotics, smart cities) by dynamically composing (i.e., chaining) application and network services (e.g., robot balancers, traffic accelerators) deployed as virtual functions (VFs) in distributed micro-clouds located at the network Edge. However, the heterogeneity of the infrastructures, the high dynamicity of services and the geographical distribution of VFs pose new challenges in terms of resource control/management capabilities, adaptive usage of multi-technology resources, and fulfillment of end-to-end latency requirements considering the impact of both processing and network delays [2][3].

In this work we present a latency-aware and reliable end-to-end service chaining mechanism based on multi-domain/multi-technology resource orchestration over geographically distributed Edge clouds interconnected through SDN. More specifically, the service chaining orchestration system exploits heterogeneous and enhanced resource control/management capabilities offered by cloud and SDN network domains to dynamically provide service chains while (i) optimally selecting VFs over the path that minimizes the offered end-to-end latency across cloud and network domains, and (ii) promptly adapting established service chain paths to the current network load. Optimal VF selection and dynamic path adaptation is triggered by real-time monitoring data collected from the underlying cloud and network infrastructures.

The proposed approach advances the state of the art since it considers both cloud processing delays and network delays while addressing latency requirements [4][5], it handles ordered sequences of VFs [6], and it also provides a service chain path orchestration feature at inter-DC level [7]. In addition, the presented mechanism goes beyond the coordinated establishment of cloud and network services, including also adaptation actions with respect to current network resource availability. In terms of experimentation, this work advances the state of the art [8][9][10] since it performs experiments in service chaining using a testbed which realistically reproduces a composite and distributed SDN/NFV deployment, including inter-DC capabilities without emulated data plane functions.

The main contribution of this work is the validation and the experimental evaluation of the proposed end-to-end service chaining orchestration system on testbed facilities reproducing a composite and distributed SDN/NFV deployment. Indeed, the experiments were performed on top of the Fed4FIRE infrastructure provided within the Fed4FIRE+ Horizon 2020 Project, which offers a federation of open, accessible and high-available Next Generation Internet (NGI) testbeds to support a wide variety of different research and innovation activities, including 5G-related experiments [11]. Taking advantage of bare metal services provided by the Fed4FIRE platform, we could benefit of exclusive use of physical servers and deploy and assess, in a realistic environment, a distributed SDN- and NFV-based systems with related resource control and orchestration functionality. We integrated the single elements of the end-to-end service chaining orchestration system that were previously presented separately and evaluated through simulations or small-scale laboratory testbeds [12][13][14].

II. Orchestration System Deployment

Figure 1 describes the architecture of our service chaining orchestration system and its deployment on top of the Fed4FIRE+ experimentation platform. According to the ETSI NFV MANagement and Orchestration (MANO) framework [15] the orchestration system is devised to run on top of an NFV Infrastructure (NFVI), which in our case was composed of three experiment slices that implemented Edge cloud SDN-based Data Center (DC) domains (named DC-1, DC-2 and DC-3), and a slice that implemented a Wide Area Network (WAN) SDN infrastructure domain providing inter-DC connectivity. An additional slice was dedicated to the Chain Optimizer. In line with [8], the proposed orchestration system is assumed to supplement NFV Orchestrator (NFVO) and VNF

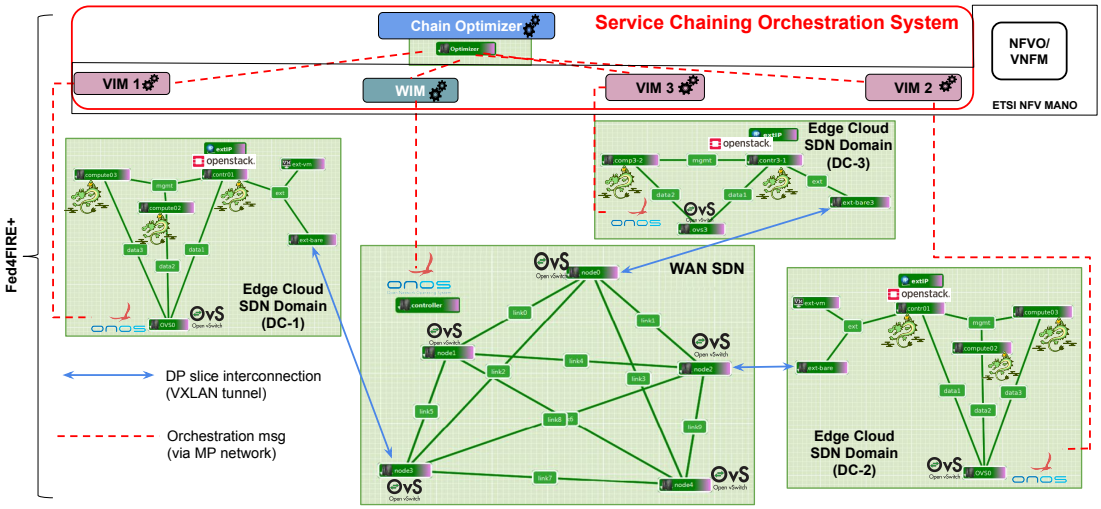

Fig. 1. Orchestration system deployment on the Fed4FIRE platform. Multiple experiment slices are interconnected at both data plane (blue/solid lines) and orchestration plane (red/dashed lines) levels.

Manager (VNFM) functionalities currently envisioned in [15] with dynamic service chaining capabilities.

The ***Chain Optimizer*** is a service chaining engine running an optimization algorithm that, upon a service chain request, selects VF instances available from different DCs to minimize an estimated end-to-end latency calculated considering both VF processing delays and inter-DC network delays information. The algorithm has been discussed in a previous work [12], where the optimization problem has been formulated as a Resource Constrained Shortest Path problem on an auxiliary layered graph accordingly defined. The algorithm uses up-to-date topological and latency information, retrieved by periodically interacting with NFVI and VF monitoring APIs to collect measurements about inter-DC latency, types and instances of VF deployed at each DCs and related processing latency. In this implementation, processing latency measurements are posted by VF instances onto the "Gnocchi" Time Series Database service provided by OpenStack on each DC slice and exposed through Gnocchi REST APIs. According to the solutions provided by the optimization algorithm, this component interacts with underlying WAN/DC domain resource orchestrators in order to enforce traffic steering operations to set-up service chain paths and, in general, to manage the lifecycle of a service chain (i.e. creation, updating and deletion). The chain optimizer leverages intent-based REST APIs exposed by underlying domain orchestrators to send such instructions using an application-oriented semantic rather than dealing with technology-specific low-level network details. The Chain Optimizer has been implemented as a Java application and it offers a REST API for CRUD (Create, Read, Update, Delete) operations on service chains.

The ***SDN WAN slice*** includes the following components. The SDN network topology consists of five physical nodes running the Open vSwitch (OvS) software, controlled by an instance of the Open Network Operating System (ONOS) controller, hosted on a dedicated node. The ***WAN Infrastructure Manager (WIM) Orchestrator*** implements the orchestration logic for the WAN SDN domain slice on top of the ONOS controller, exposing the programmable provision of service chain paths across the WAN by means of an intent-based northbound REST interface. Hence, the set-up of service chain paths in the WAN to connect VFs in different DCs can be triggered by the Chain Optimizer by specifying to the WIM orchestrator the list of DCs to be traversed. Then, the WIM orchestrator derives the DC domain gateways to be connected and performs mapping operations by identifying the network path and, accordingly, enforces the forwarding rules to the switches along the identified path. In line with [16], the WIM orchestrator also offers reliable service chains by adapting (i.e., redirecting) service paths, or a segment thereof, to recover from network congestions events detected by periodically collecting statistics from the SDN controller and deriving up-to-date switch link throughput data. Finally, the WIM is also responsible for the collection of network latency information (i.e., inter-DC delays) that are made available to the Chain Optimizer for computing a minimum-latency service graph. Details on WIM Orchestrator can be found in [13][17].

The three ***SDN DC slices*** host small Edge cloud deployments based on OpenStack. Each DC slice includes two or three compute nodes, where virtual machine instances are deployed over a QEMU-KVM hypervisor. All OpenStack nodes are connected to another physical node running an instance of OvS, representing the data plane SDN infrastructure of the DC, which is controlled by an instance of ONOS running locally. The same physical node hosts also the ***Virtual Infrastructure Manager (VIM) Orchestrator***, which implements an SDN-enabled DC/cloud domain orchestration logic providing advanced network management capabilities in cloud computing environments. The VIM orchestrator exposes an intent-based northbound REST interface that allows to specify a service chain by means of a high-level descriptive syntax, agnostic to the specific SDN technology adopted [14]. This makes it suitable to manage different DC domains in a multi-technology environment, e.g., leveraging different SDN controllers. The

VIM orchestrator is also capable of dynamically applying changes to an existing service chain without having to delete and re-deploy it from scratch. This allows to dynamically adapt service chains to the current context of users or services (e.g., location of users in a mobility scenario) or to varying needs of the service provider (e.g., resource management policy), and, ultimately, to avoid or prevent SLA violations. Furthermore, the REST API provided by the VIM orchestrator allows the Chain Optimizer to collect information about the currently deployed VFs and their estimated processing latency, computed based on the current workload. The details of the VIM orchestrator can be found in [18][14].

The established DC slices and the WAN slice interact at the data plane level by exchanging packet data traffic by means of VXLAN tunnels, and at the orchestration plane level by exchanging control messages between Chain Optimizer, WIM and VIM orchestrators. Interactions at the network control plane level do not take place between different slices. This is in line with the envisioned architecture, where each domain is supposed to adopt its own SDN control plane solution independently of the choice made by other domains.

III. Experimental Results

In this section we present experimental results to validate the correct operations of our orchestration system, as well as some performance evaluation of the software components. We generated *create* and *delete* service chain requests to the Chain Optimizer, with different lengths and requirements in terms of bandwidth and maximum latency. The Chain Optimizer handles each request, computes a latency-optimized solution and sends the corresponding forwarding instructions to the relevant VIM and WIM orchestrators through their respective northbound interfaces. Then, each VIM/WIM interacts with the SDN controller in its domain in order to setup the relevant flow entries. After the switches have been configured and the chain is correctly established, we inject data traffic across the VF instances implementing the chain (e.g., by generating iperf flows with a bit rate equal to 1 Mbit/s).

Figure 2 shows the sequential time diagram of the throughput measured at the VF instances involved in the deployment of the following service chain sequence: i) VF-1 $\rightarrow$ VF-7 $\rightarrow$ VF-9, ii) VF-1 $\rightarrow$ VF-9, iii) VF-1. At time $t = 0$ the three chains have already been successfully deployed. According to the initial placement, instances of VF-1 and VF-7 are deployed in DC-1, whereas instances of VF-9 are deployed in DC-2. We can observe that at $t = 9$s traffic starts flowing through instances involved in the first chain, i.e., VF-1, VF-7 and VF-9 (throughput equal to 1 Mbit/s). When traffic is sent through the second chain, at $t = 40$s a second flow is measured at VF-1 and VF-9 instances (throughput equal to 2 Mbit/s). Finally, when the third chain is loaded with traffic, throughput equal to 3 Mbit/s is measured at VF-1 instance at $t = 70$s. At the end of the experiment ($t = 100$s), the measured throughput drops to zero due to the deletion of the three service chains. This demonstrates the correct deployment and deletion of the service chains across the involved domains.

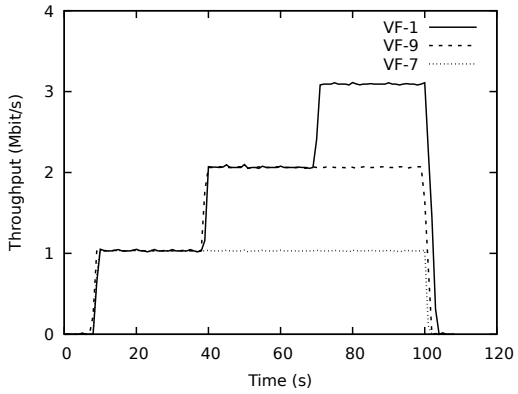

Fig. 2. Sequential time diagram of the throughput measured at VF instances involved in a service chain deployment sequence.

Then, we considered 4 different sets of 10 requests with service chains having the same length (from 2 to 5), sent to the Chain Optimizer with an inter-arrival time equal to 60s. We computed the overall response time, which is the time interval, measured at the Chain Optimizer side, between the reception of a request and the delivery of a response to the client. In case of successful request, this interval includes also the time needed for sending the forwarding instructions to VIM and WIM orchestrators and receiving their reply. Table I reports the measured values as a function of the service chains length. As expected, the overall time increases when the chain length increases since the WIM and VIM orchestrators require more time for the chains installation. Moreover, we measured the CO computation time as the time needed by the VNF Selection algorithm for solving the optimization problem. It remains almost stable (around 2-3ms) due to the system scale that in an experimental set-up remains small anyway.

Table I also reports the average response times of the WIM and VIM orchestrators, corresponding to the interval between the reception of a request from the Chain Optimizer and the delivery of a response. In the WIM case, it includes the search for an available path and the setup of the flow entries in the OpenFlow switches, which explains the relative high values (around 40s). Measured values are lower in the VIM case (around 1.5s), because they include only the response times of the REST interfaces of the underlying SDN controllers, without verifying the actual setup of the flow entries in the data plane. The reason for this choice is to measure the VIM response time independently of the SDN controllers used in the DC domains, which can operate their southbound interfaces in different ways.

TABLE I
Performance of the orchestration system components.

Chain Length	CO Overall Resp. Time [s]	Resp. Time WIM [s]	Resp. Time VIMs [s]
2	64.34	40.84	1.41
3	69.34	36.29	1.50
4	74.96	35.98	1.48
5	82.75	40.5	1.52

The adaptation feature offered by the orchestration system with respect to the network status in the SDN WAN has also been tested. Indeed, the WIM orchestrator adapts the network paths across the WAN underpinning the VF chain path segments with respect to the load of switches/links derived from a selective set of monitoring data (e.g., data throughput). If one or more switches become overloaded (the threshold is fixed to 1 Gbit/s), it triggers the redirection of service chain data flows through other available switches thereby preventing or recovering from service degradations due to switch congestions. Thus, we measured the redirection time, i.e., the time required by the WIM orchestrator to look for a new path and install the relevant flows, and to delete flows from the overloaded switches along the old path. Measurements showed that the time required to setup a path triggered by the WIM orchestrator is (generally) influenced by the number of edge cloud domains traversed by the chain being established, e.g.: setup time is around 20s when a chain is spread across 2 DCs, and around 30-35s for chains spreading across 3 DCs. On the other hand, the measured redirection time was relatively low (around 6s) with respect to the overall setup time.

Table II compares the end-to-end latency obtained as the sum of retrieved VFs processing latency and the inter-DC latency measurements, used by the Chain Optimizer to compute the service chain path (i.e., e2e latency @Chain Optimizer), with the end-to-end latency actually experienced by data while flowing in the established service chains and measured using ping commands (i.e., e2e latency @established chains). For each run a sequence of 10 service chain requests is sent for a given chain length. We can notice that the actual measured values and the ones estimated by the CO are pretty close, which proves the robustness of the latency-awareness feature and of the computation process of our orchestration system that allows for an efficient selection of the DCs and of the VFs instances.

TABLE II
END-TO-END LATECY: @CHAIN OPTIMIZER VS. @ESTBLISHED CHAINS.

Chain Length	Latency @Chain Optimizer [s]	Latency @established chains [s]
2	56.24	76.7
3	72.38	77.2
4	103.86	112

As part of a side-activity assessment, we measured the average TCP throughput when the generated traffic traverses service chains with an increasing number of VF instances. As a result, the average throughput was very close to the maximum value achievable with the 1 Gbps physical interfaces installed in the Virtual Wall servers. The measurements we obtained proved that the Virtual Wall facility is able to provide full capacity to a typical NFV/SDN infrastructure based on OpenStack and Open vSwitch components. This is mainly due to the possibility offered by Virtual Wall to deploy slices using bare-metal servers, thus avoiding the overhead of an infrastructure emulated through, e.g., nested virtualization. Therefore, the Fed4FIRE+ facilities (and Virtual Wall in particular) can be considered a good candidate to perform realistic experiments on non-trivial NFV/SDN infrastructures based on production-level software tools.

IV. CONCLUSIONS

In this work we presented an end-to-end orchestration system comprising dynamic VF selection and intent-based traffic steering control capabilities supporting latency-aware and reliable service chaining over geographically distributed SDN-based cloud DCs interconnected through SDN WAN. We carried out experiments to validate the orchestration system using the Fed4FIRE platform that allowed us to set-up a realistic and composite SDN/NFV deployment. These activities also allowed us to derive lessons learnt and best practices to foster the use of SDN and virtual infrastructure and effectively enable 5G service delivery on top of distributed edge cloud deployments. In the near future, we plan to integrate the proposed orchestration functionalities with a complete implementation of a MANO orchestrator, as well as perform extensive evaluation in comparison with alternative VF selection approaches.

ACKNOWLEDGEMENT

This work was funded in part by the LASH-5G project (Grant Agreement no. 732638) and the DiMoViS-TRIANGLE project (Grant Agreement no. 688712),

REFERENCES

[1] F. Yousaf *et al.*, "NFV and sdn key technology enablers for 5g networks," in *IEEE Journal on Selected Areas in Communications*, 2017.
[2] S. Zhang *et al.*, "5G: Towards energy-efficient, low-latency and high-reliable communications networks," in *IEEE ICCS*, Nov 2014.
[3] I. Parvez *et al.*, "A survey on low latency towards 5G: Ran, core network and caching solutions," *IEEE Communications Surveys Tutorials*, 2018.
[4] A. M. Medhat *et al.*, "Near optimal service function path instantiation in a multi-datacenter environment," in *IEEE CNSM*, 2015.
[5] A. Abujoda and P. Papadimitriou, "Midas: Middlebox discovery and selection for on-path flow processing," in *COMSNETS*, 2015.
[6] A. Lombardo *et al.*, "An analytical tool for performance evaluation of software defined networking services," in *IEEE NOMS 2014*.
[7] M. T. Thai *et al.*, "A joint network and server load balancing algorithm for chaining virtualized network functions," in *IEEE ICC*, 2016.
[8] A. M. Medhat *et al.*, "Extensible framework for elastic orchestration of service function chains in 5g networks," in *IEEE NFV-SDN*, Nov 2017.
[9] B. Sonkoly *et al.*, "UNIFYing cloud and carrier network resources: An architectural view," in *IEEE GLOBECOM*, 2015.
[10] A. Sgambelluri *et al.*, "Orchestration of Network Services Across Multiple Operators: The 5G Exchange Prototype," in *EUCNC*, 2017.
[11] https://www.fed4fire.eu/.
[12] B. Martini *et al.*, "Latency-aware composition of virtual functions in 5g," in *Proceedings of IEEE NetSoft 2015*, April 2015, pp. 1–6.
[13] A. Mohammed *et al.*, "Sdn controller for network-aware adaptive orchestration in dynamic service chaining," in *IEEE Netsoft 2016*.
[14] F. Callegati *et al.*, "Performance of intent-based virtualized network infrastructure management," in *Proceedings of IEEE ICC 2017*.
[15] "Network functions virtualisation (NFV): Architectural framework," *ETSI GS NFV*, vol. 2, no. 2, p. V1, 2013.
[16] Y. Boucadair *et al.*, "Service function chaining service, subscriber and host identification use cases and metadata," IETF Secretariat, Tech. Rep. draft-sarikaya-sfc-hostid-serviceheader-04.txt, 2017.
[17] B. Martini and F. Paganelli, "A service-oriented approach for dynamic chaining of virtual network functions over multi-provider software-defined networks," *Future Internet*, vol. 8, no. 2, p. 24, 2016.
[18] F. Callegati *et al.*, "Sdn for dynamic nfv deployment," *IEEE Communications Magazine*, vol. 54, no. 10, pp. 89–95, 2016.