Network Service Chaining using Segment Routing in Multi-Layer Networks

Francesco Paolucci

***Abstract*—Network Service Chaining, originally conceived in the Network Function Virtualization (NFV) framework for Software Defined Networks (SDN), is becoming an attractive solution for enabling service differentiation enforcement to microflows generated by data centers, 5G fronthaul and Internet of Things (IoT) cloud/fog nodes and traversing a metro-core network.**

However, the current IP/MPLS-over optical multi-layer network is practically unable to provide such service chain enforcement. First, MPLS granularity prevent microflows to be conveyed in dedicated paths. Second, service configuration for huge number of selected flows with different requirements is prone to scalability concerns, even considering the deployment of a SDN network.

In this paper, effective service chaining enforcement along Traffic Engineered (TE) paths is proposed using Segment Routing and extended traffic steering mechanisms for microflows mapping. The proposed control architecture is based on an extended SDN controller encompassing a Stateful Path Computation Element (PCE) handling microflow computation and placement supporting service chain, whereas segment routing allows automatic service enforcement without the need of continuous configuration of the service node. The proposed solution is experimentally evaluated in a Segment Routing over Elastic Optical Network (EON) network testbed with a deep packet inspection service supporting dynamic and automatic flow enforcement using Border Gateway Protocol with Flow Specification (BGP Flowspec) and OpenFlow protocols as alternative traffic steering enablers. Scalability of flow computation, placement and steering is also evaluated showing the effectiveness of the proposed solution.

***Index Terms*—Network Service Chaining, Software Defined Networking, Multi-Layer Networks, Open Database, Segment Routing, LSP, Path Computation Element, Border Gateway Protocol, OpenFlow Protocol, flow steering.**

I. Introduction

In next-generation metro and core networks, operators will be required to transport different application traffic, from/to specific client networks (e.g., data centers, 5G Radio Access Networks (RAN) [2], smart Internet of Things (IoT) cloud/fog nodes processing data from/to massively distributed sensors [3], [4], blockchain-based platforms [5], [6]) where each application may generate an huge number of low or medium bitrate flows (i.e., MicroFlows) subject to different end-to-end Quality of Service (QoS) requirements [7]. Moreover, specific MicroFlows, in order to satisfy the requested QoS and additional requirements such as traffic isolation, security and performance monitoring, may also be required to traverse single or combined network functions (e.g., firewall, deep packet inspection, policers, accelerators) before reaching the destination. The dynamic enforcement of combined functions to selected flows, called Network Service Chaining (NSC), is a hot topic in the context of Software Defined Networking (SDN) [8], [9]. Service chaining concepts and deployments have known a rapid success in the context of Virtual Function Networking (NFV) framework, for which the setup of cascaded services is typically deployed in virtual environments (e.g. virtual machines, containers) and are distributed and connected across a cloud infrastructure by a dedicated hypervisor or NFV orchestrator.

In the case of network services made available in the metro and core network directly (e.g., through dedicated hardware, middleboxes or edge/fog computing platforms co-located with core metro/core nodes), efficient NSC deployment is nowadays limited by the current multi-layer architecture supporting IP/MPLS in the packet-switching capability network layer. Indeed, the dynamic enforcement of service chaining is limited by the Label Switched Paths (LSP) granularity thus preventing specific per-flow service differentiation.

MicroFlows originated in the access/aggregation network segment are not mapped onto core LSPs in a 1:1 match for scalability reasons (i.e., per-MicroFlow signaling is not considered a realistic and feasible solution). Moreover, the pure SDN framework would require a parallel configuration of all the nodes involved in the microflow path computation, including those providing network services. If such approach is considered effective in the access, in local area networks or in intra-data center scenarios, this is completely unrealistic in a metro-core network due to scalability issues. Therefore, the selection of one or more existing LSPs at the edge of the core network is required when a new MicroFlow has to be served. Such selection (i.e., flow computation) requires specific algorithms to address network utilization performance, scalability and fast steering enforcement. This steering operation is currently based on local and manual policies (e.g., enforced by means of command line interface at the edge network elements). Nonetheless, also considering standard MPLS transport, automatic service chain enforcement without the need of continuous and potentially unscalable SDN configuration of middleboxes or edge/fog nodes remains an issue.

In this work, an integrated SDN control architecture enabling an effective computation, deployment and control of service chained microflows is proposed in the framework of the Stateful Path Computation Element PCE for metro-core multi-layer networks architecture exploiting network information databases. The proposed solution exploits Segment Routing (SR)-based multi-layer networks [10], [11]. Segment Routing is a Traffic Engineering (TE) technique proposed by the Internet Engineering Task Force (IETF) that significantly simplifies the control plane operation and natively supports service chaining enforcement without the need of dynamically configuring the service node. The architectural solution includes a novel flow computation element and two microflow traffic steering mechanisms, the

Manuscript received March 28, 2018.

F. Paolucci is with Scuola Superiore Sant'Anna, Pisa, Italy.

This paper is an extended version of the work presented in [1]. This work has been partially supported by the EU H2020-ICT-14 project 5GEx (grant n. 671636). Special thanks to Alessio Giorgetti for the significant effort spent in the OpenFlow testbed configuration.

former compatible with legacy equipments exploiting Flow specification extensions for Border Gateway Protocol (BGP FlowSpec) [12], the latter suitable for pure SDN-capable edge nodes exploiting OpenFlow protocol. The proposed architecture has been evaluated in an experimental multi-layer testbed. In particular, a deep packet inspection (DPI) service node has been implemented and validated in the testbed, along with steering procedures at the edge node without requiring further control configurations at intermediate nodes and service nodes. Extensive measurements of flow computation and steering enforcement procedures have been carried out to validate the feasibility and the scalability of the proposed approach.

With respect to our preliminary work [1], this work, besides related work analysis, extends and refines the architectural aspects, introduces alternative traffic steering mechanisms and provides overall experimental results with focus on microflow computation and placement scalability.

II. Related work

Historically conceived in parallel with the SDN/NFV framework, most of the literature works in service chaining focuses on intra-data center scenario and targets SDN-controlled layer-2 networks.

Work in [13] targets the placement and the composition of security-based VNFs for apps requiring security treatment differentiation in intra-data center networks. In particular the work in [14] targets the optimal placement of NSC middleboxes encompassing function dependencies and chain correlations considering the end-to-end delay and the bandwidth consumption as performance metrics.

Work in [15] proposes and implements an SDN orchestrator with service chaining composition capabilities and adaptive provisioning of delivery paths. The implementation includes a northbound interface based on Representational State Transfer (REST) Application Programming Interface (API) exposing service deployment functions to higher-level service orchestrator/hypervisor or applications. Service chain enforcement is realized by standard OpenFlow-based southbound interfaces.

Works in [16] and [17] aim at minimizing optical/electronic/optical conversions in a data center multi-layer infrastructure in order to efficiently deploy and connect virtualized network functions. Optimization is proposed by properly grouping network functions within the same DC packet-switched island, called performance optimized data center (POD), such that inter-POD communications through the optical layer are minimized. The proposed architecture includes intra-DC traffic steering performed by means of aggregated MPLS flows or OpenFlow-based packet rule enforcement.

With focus on integrated inter-data center and multi-layer connectivity, the work in [18] provides an overall network function virtualization and orchestration architecture, moving the SDN control of a virtual tenant network in the cloud and assessing SDN/NFV orchestration with different independent SDN control instantiations over a common multi-layer infrastructure. In this context, control plane functions, e.g., PCE have also proposed to be virtualized to guarantee replication service, improved synchronization and reliability [19]. The recent work in [20] extends the applicability to the 5G architecture. Virtual backhaul tenants with both virtual control and infrastructure are deployed connecting RAN and their virtual Evolved Packet Core (vEPC) function residing in the cloud.

The segment routing forwarding mechanism has been investigated and assessed in multi-layer networks, drastically simplifying control plane operation. SR enables TE by enforcing label-stack routing only at source node, thus avoiding complex and time-consuming signalling procedures, either distributed, e.g., by the Reservation Protocol with TE extensions (RSVP-TE), or triggered by a central controller, e.g., in SDN networks using OpenFlow or NETCONF. Algorithms have been proposed to effectively minimize the network path segment list depth while addressing equal cost multi-path selection [21] [22], thus allowing SR for service chain utilization, which implies an increase of the average segment list depth. The work in [10] showed an experimental assessment in a multi-layer network including the PCE. The work in [11] validates SR reliability mechanisms and extends the applicability of SR to multi-domain network scenarios. Furthermore, the SR technology has been exploited to provide real-time Service Level Agreement information feedback to orchestrators deploying network services with strict latency constraints [23].

The state-of-the-art of traffic steering solutions including the service chaining support, have been proposed mainly in the context of intra-DC scenarios and targeting layer-2 networks. A detailed proposal allowing service chaining in a layer-2 network was presented in the context of the stEERING framework [24], in which an enhanced SDN controller performs both OpenFlow-based traffic steering towards pre-programmed service chains. Furthermore, segmented service chain configurations of middleboxes have been proposed for a layer-2 network [25].

All these works do not consider a coordinated architecture and an enabling technology suitable for multi-layer metro-core network. Moreover, they do not address the scalability issues of SDN-oriented network configuration, especially related to middleboxes enforcement due to each microflow computation and steering. This work goes beyond the existing state-of-the-art by proposing a complete control architecture enabling efficient service chain exploiting SR over EON. In particular, the work proposes an extended controller with a novel flow computation element, providing full computation and steering workflow. Moreover, it proposes two steering techniques extending BGP Flowspec and OpenFlow enforcing flow service chain without the need to directly configure service nodes. Finally, this work provide extensive experimental scalability evaluation demonstrating the feasibility of the proposed approach.

III. Service Chaining with Segment Routing: Architecture

The proposed architecture enabling effective NSC and microflow computation and mapping enforcement exploiting the SR-based multi-layer network is shown in Fig. 1. A core/transport network connects different client networks, hosting heterogeneous traffic flow originators, such as data centers or fog nodes. A number of network functions are available directly in the metro-core network as middlebox devices, offering different or combined service treatment to specific traffic injected in the core. For example, a deep packet inspection function (service S1) is implemented by

node E in Fig. 1. Several applications run in the attached client networks generating traffic (microflow) requests. In particular, microflow requests may be referred either to end-to-end tunnelled traffic (e.g., deployed from network-sliced orchestration based on NFV-enabled infrastructure, thus identified with a MPLS label or Pseudowire/Virtual Routing and Forwarding tags) or to general application-specific traffic (e.g., video-on-demand).

A. Segment Routing and Stateful PCE

The core/transport multi-layer network is assumed to run control plane based on the SDN framework. In this scenario, a NSC network controller is in charge of orchestrating connection setup/release/modification events by storing and updating the network topology, the TE database and the state of the established connections. For such reason, an active Stateful PCE [26] is assumed to run inside the multi-layer network controller.

In the use-case of Fig. 1, the metro-core network domain is based on Segment Routing (SR) and the Stateful SR-PCE inside the NSC Controller computes, establishes and releases SR Label Switched Paths (SR-LSPs). Indeed, in the SR case, TE does not require the utilization of signaling messages [10]. Specifically, a stack of MPLS labels (i.e., the SR segment list) is enforced at the edge node of the multi-layer network domain to each packet belonging to a given Flow to enforce explicit routing. Moreover, SR enables effective NSC by associating each available network service to a special MPLS label (i.e., service label) to be included in the segment list in the case the network service activation is desired for the traffic conveyed in the SR-LSP. This way, intermediate nodes and middleboxes (e.g., node E in Fig. 1) do not require specific configurations for NSC. Service chain including more service functions inside the network is exploited by assigning a segment list including all the service labels associated to the required functions with the desired order. This way, the computed segment list represents the joint network-service route.

In this scenario, the LSP-DB at the SR-PCE stores the set of the segment lists (i.e., the SR-LSPs) currently used in the network. Stateful condition of the PCE is a stringent requirement, since bandwidth requests are not signalled in the data plane, but only reserved in the TED and LSP-DB databases of the PCE. Each SR-LSP LSP_i entry is stored in the LSP-DB (of size L) with a pre-planned reserved bandwidth value B_i (where $0 \leq i < L$ is the LSP-DB entry index). The same bandwidth value is reserved for each traversed link in the PCE TED. The service-chained flow f_k (where $0 \leq k < K$ and K is the number of installed flows in the network) requiring to traverse the core network will be associated to an existing SR-LSP (entry i in the LSP-DB) and its required bandwidth b_k^i will consume part of the B_i SR-LSP bandwidth. In the case new flow requests cannot identify an existing SR-LSP due to bandwidth bottleneck, the stateful PCE will instantiate a new SR-LSP in the control plane or it will trigger lightpath activation (or EON existing lightpath elastic operation [27]) in the optical layer to export new links and reservable bandwidths in the TED, by means of southbound SDN-based interfaces. For example, OpenFlow [28] or NETCONF/YANG [29] interfaces may be utilized to trigger parallel configuration of all the optical nodes identified by the lightpath route without the need of a signaling protocol. In particular, in transparent EON, flow entries enable signal cross-connection and spectrum filtering enforcement (in EON, flexible grid with frequency slots multiple of 12.5GHz is employed), whereas additional flow entries at edge nodes configure the signal tx/rx physical parameters (e.g., transceiver type, modulation, code) [30], including sliceable transponders and super-channels [31].

B. Flow Computation Element and Flow Steering API

In the NSC Controller, besides the SR-PCE, two novel modules are introduced in charge of receiving, computing and enforcing service chained microflow. The former, namely the Flow Computation Element (FCE) module, is in charge of receiving and computing microflow requests. The latter, namely the Flow Steering API, communicates with the source edge node providing the segment list or the ID of the segment list to be enforced to the requested Flow, thus performing traffic steering. The flow request is identified by a flow match, a requested bandwidth value b and the type/chain of services needed for that flow. When a new flow request is submitted, the FCE module performs flow computation, i.e., it identifies one or more SR-LSPs established in the control plane over which to steer the Flow. If the new Flow cannot be served using active LSPs, one or more SR-LSPs are properly initiated by SR-PCE at the considered edge node using either the PCE Protocol with instantiation capabilities [32] or the OpenFlow protocol. The FCE module resorts to four databases to perform the Flows mapping:

1) the PCE Routing Information Base (RIB), queried to identify the edge nodes associated to traffic source and destination;
2) the SR-LSP database (LSP-DB), queried to retrieve the active SR-LSPs that match the Flow constraints, edge points and required network service chain;
3) the Traffic Engineering Database (TED), to compute the path for possible new SR-LSPs;
4) the Flow database, storing all the installed flows with reserved bandwidth and the related hosting LSPs.

Since the number K of installed Flows in the metro-core network may be huge, a Flow database is proposed to be stored in an external Open Network Database [33] suitable for application-based control and service plane. Such databases have been conceived in order to store TE (i.e., topology and per-link reserved bandwidth) and stateful objects (i.e., active connections) of different layers (i.e., lightpaths, TE links, SR-LSPs) and the correlation among them, exploiting the scalability performances of the latest generation of databases technologies. The flow database stores the flows currently installed in the network. Each flow entry includes the flow attributes, the bandwidth reserved for the flow and the hosting SR-LSP.

Fig. 2 shows the flow computation and steering workflow. First, the new flow request (identified by source/destination nodes, required bandwidth, requested service chain) is processed in order to retrieve the source and destination edge nodes (i.e., the data center gateways) by querying the RIB database. Then, candidate SR-LSPs are selected in the LSP-DB satisfying the requested service chain. In the case of no candidates, a request will be issued to the Stateful SR-PCE in order to setup a new SR-LSP with a reserved overall capacity B_i. Among the set $Scand$ of candidates (of size $W \leq L$, and index $0 \leq w < W$), if $W > 1$, specific policies are applied

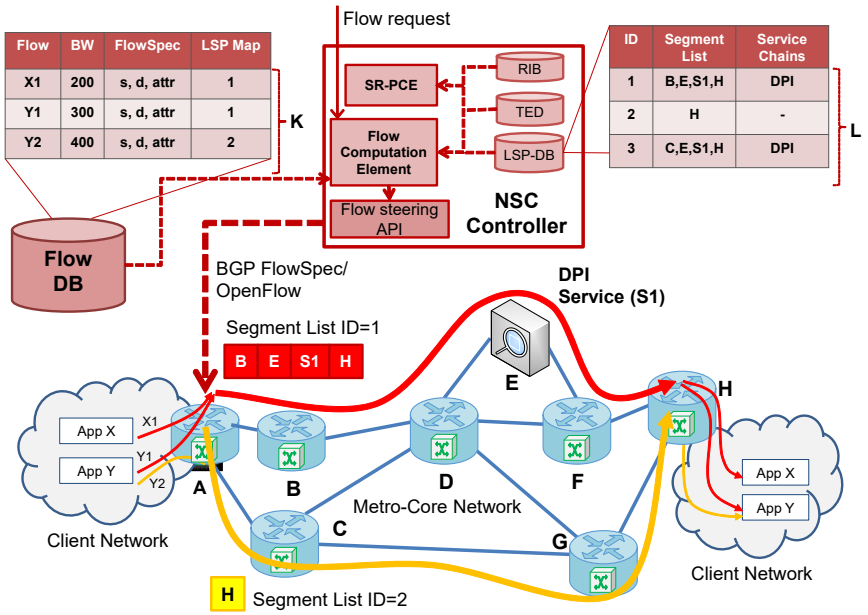

Fig. 1. Segment Routing based Network architecture enabling service chaining.

to find the most suitable LSP. For example, least congested policy may be run by computing the current LSP_w amount of bandwidth $B_w^{cur} = \sum_{k \in S^w} b_k^w$ (where S^w denotes the subset of flows f_k carried out by the only LSP LSP_w) consumed by all the flows carried out by the candidate SR-LSP and thus, obtaining the available bandwidth $B_w^{av} = B_w - B_w^{cur}$. Finally, the least congested candidate LSP is identified for which $\max_{S_{cand}} B_w^{av} - b$ is verified. To perform that, the Flow database and the LSP-DB need to be queried in order to retrieve, for each candidate SR-LSP LSP_w the b_k^w and the B_w values, respectively.

The described control scheme also enables TE solutions. In Fig. 1, two segment lists (i.e., SR-LSPs) are used between edge nodes A and H, each one allowing 1000 bandwidth units. The segment list B-E-$S1$-H has ID=1 and specifies the red SR-LSP including the execution of the DPI service (service label $S1$) at node E. The segment list H has ID=2 and is composed only by the destination node identifier (i.e., the A-H shortest path), thus identifying the yellow SR-LSP. Moreover, three Flows generated by two applications (i.e., app X and Y) are already established, as shown in the Flow DB table, and associated with one of the SR-LSPs. When a new Flow arrives requiring DPI treatment, FCE selects SR-LSP with ID=1. However, if requested bandwidth exceeds 500 bandwidth units, the FCE should compute a new segment list, for instance C-E-$S1$-H with ID=3 enforcing path A-C-D-E-F-H, e.g. to avoid the bottleneck link B-D, and assuring the DPI service at node E. In the case DPI service is not required, the segment list with ID=2 is selected.

IV. Extended Traffic Steering Mechanisms

After flow computation, if a hosting SR-LSP is identified, the flow request is stored in the Flow database. The flow entry in the database includes the hosting LSP_i allowing flow computation for future requests. Then, actual Flow steering is enforced by means of Flow steering API, a southbound interface which may be dedicated to flow instantiation or shared by other control protocols (e.g., for tunnel configurations). The selected API depends on the controlled network technology. In the case of legacy MPLS networks existing protocols such as BGP may be exploited using the Flowspec extensions proposed in Sec. IV-A, while in case of pure SDN networks, SDN-based protocols such as OpenFlow may be exploited, as detailed in Sec. IV-B.

A. Extended BGP Flowspec steering

Flow specification extensions for Border Gateway Protocol (BGP FlowSpec) have been proposed in order to match a flow based on packet attributes and enforce it to specific actions, mainly addressing security issues [12]. Recent extensions enable the of a specific action to be enforced to the flow identified by BGP FlowSpec match fields.

The proposed steering procedure conceives a BGP speaker located at the NSC Controller as southbound interface that collects the outputs of flow computation and sends one or more UPDATE messages only to the ingress edge node of the selected SR-LSPs. The UPDATE message encloses the information required to univocally identify the Flow, i.e., the FlowSpec match attributes (e.g., source/destination IP address, IP protocol, layer4 ports, Differentiated Service Code Point - DSCP) and the related actions to be applied to the specific Flow. The main advantage of this technique is that BGP is already widely utilized and available in most core network equipment, thus core edge router can perform steering directly by processing BGP FlowSpec UPDATE messages sent by the NSC Controller.

In order to disaggregate the steering enforcement from the mechanism utilized to instantiate the SR-LSP in the ingress node (it is supposed that the SR-LSP has been previously configured in the ingress node equipped with the computed segment list) we propose to extend BGP FlowSpec with three novel actions to create, remove and modify the Flow steering into existing SR-LSPs. Each action is encoded with three fields enclosing:

1) the action type (i.e., create, modify, remove)

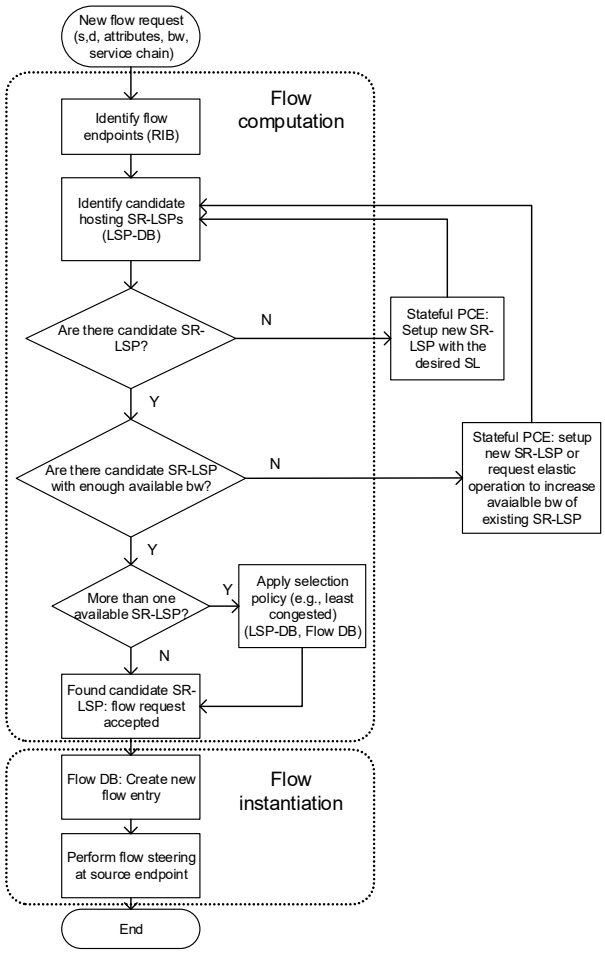

Fig. 2. Flow Computation Element and Flow Steering API: flow computation and steering workflow.

2) the LSP identificator into which the Flow has to be steered;
3) the indication of the protocol instantiating the SR-LSP (e.g., PCEP, NETCONF, OpenFlow, MPLS, Segment Routing), allowing the steering independently from the protocol (and the technology) utilized to instantiate the LSP.

B. OpenFlow-based steering

In the case the edge node is a SDN-capable node (e.g., edge OpenFlow switch) the traffic steering mechanism may be enforced via OpenFlow protocol by configuring flow and group tables of the switch. Similarly with respect to BGP Flowspec, the SR-LSP is first installed in the ingress switch by means of a `GROUP_MOD` message, specifying the computed segment list, i.e. the MPLS stacked labels.

In this specific case, the `INDIRECT` group type is utilized since it allows the aggregation of different flow matches onto a single list of actions. The purpose of an `INDIRECT` group is to consolidate common actions of a set of flows in order to reduce flow table memory consumption. Then, for each microflow steering, a `FLOW_MOD` message is generated by the controller specifying the match and the default action (i.e., application to the specified OpenFlow group). In the case of multiple flows defining a single service, a bulk of `FLOW_MOD` messages is enforced for each flow, followed by a `BARRIER_REQ` message. This allows steering enforcement to be applied in an atomic fashion. Moreover, this will trigger the switch to reply with a `BARRIER_REP` message thus acknowledging the overall traffic enforcement activation.

V. Experimental Results: Demonstration and Scalability

The proposed NSC-enabled control architecture have been evaluated in a SR multi-layer network testbed in two different configurations, exploiting the two different steering techniques and southbound APIs detailed in the previous section.

In the first configuration, shown in Fig. 3, legacy IP/MPLS over optical elastic optical network is employed and the considered data plane is composed by Juniper routers (R_x in Fig. 3) running OSPF-TE equipped with a set of agents enabling the utilization of SR [10] and Ericsson SPO 1400 ROADMs, connected by means of Gigabit Ethernet interfaces and muxponder optical card supporting 10G lightpaths.

In the second configuration, shown in Fig. 4, pure SDN over optical is employed, utilizing the same optical devices, whereas Juniper routers are replaced by OpenFlow switches (H_x in Fig. 3), implemented either with Ubuntu 16.04 Linux boxes (CPU Intel QuadCore 2GHz equipped with 4GB RAM) running Open vSwitch (OVS) 2.4 kernel mode [34] or with OpenFlow-enabled HP ProCurve 2500/2800 switches equipped with optical interfaces.

In both configurations, a Linux box server equipped with two Intel Gigabit Ethernet interfaces is included (i.e., node F in Fig. 3 and Fig. 4) implementing a simplified DPI with firewall capabilities and SR functionalities by using OVS v2.4. The DPI participates to the OSPF-TE instance announcing node F with DPI service (service label $S1$). The overall network is controlled by a NSC Controller including a SR-PCE [10] and the Flow Computation module, as described in Sec. III-B, developed in C++. The NSC Controller runs over a Linux Ubuntu 16.04 machine equipped with an Intel Xeon Quad-core CPU (3.40GHz) and 4GB RAM. Two different southbound API are employed in the two configurations. In the legacy configuration, a BGP FlowSpec Speaker module is included, developed in C++. Edge node $R1$ is equipped with an agent running PCEP and extended BGP FlowSpec. Node agent elaborates BGP FlowSpec update messages and enforce flow steering by sending interactive CLI-based scripts to edge routers. In the pure SDN configuration, the implementation has been embedded in the open source SDN Ryu controller, and the OpenFlow protocol is employed for control communication directly with the OpenFlow switches control interface. OpenFlow version 1.3 is employed. Traffic flows are injected by a Spirent SPTN4U traffic generator and analyzer connected to edge nodes $R1$ ($S1$) and $R5$ ($S5$), respectively.

For both configurations, two 10G lightpaths $lp1$ and $lp2$ are installed in the optical network, providing optical bypass and 2 TE links at routers OSPF-TE instance interconnecting $R1(H1)$-$R3(H3)$ and $R1(H1)$-$R5(H5)$, respectively. Therefore, flows requiring DPI will be sent to $lp1$ through the F-$S1$-$R5(H5)$ segment list without the need of configuring neither intermediate nodes nor the DPI server. In fact, $R1(H1)$-F shortest path is the two-hops $R1(H1)$-$R3(H3)$-F route through $lp1$ bypass). Flows not requiring DPI service will be sent to $lp2$ through the $R1(H1)$ segment list, which is the shortest path between edge nodes $R1(H1)$ and $R5(H5)$.

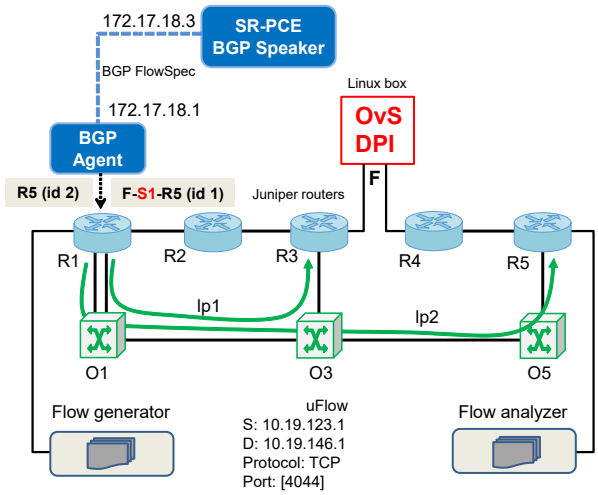

Fig. 3. Experimental testbed with legacy routers using BGP Flowspec as Flow Southbound API.

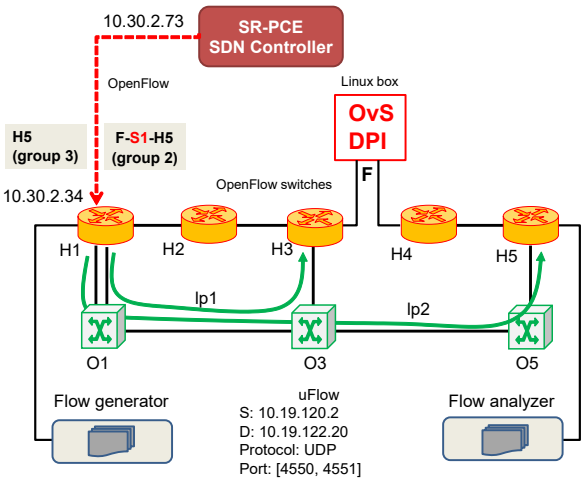

Fig. 4. Experimental testbed with SDN switches using OpenFlow as Flow Southbound API.

With reference to the the first configuration of Fig. 3, Fig. 5 shows a Wireshark capture collected at the BGP Speaker of the stateful PCE (IP address 172.17.18.3). The capture shows the `UPDATE` messages sent to the agent (IP address 172.17.18.1). In particular, one message referred to a flow requesting DPI service is expanded showing the FlowSpec `NLRI` field describing the Flow match filter (i.e., source address 10.19.123.1, destination address 10.19.146.1, protocol TCP, TCP port 4044). Moreover, the novel Extended Communities Path Attribute is shown, enforcing flow steering action to a generic path. In this case the `ACTION` field is set to *New Flow Steer*. The `PROTOCOL` field is set to *Local SR-SC list*, i.e., the hosting path id is identified within the set of segment lists including special SC labels installed locally. Finally, the installed path identifier (i.e., `INSTALLED PATH ID` = 1) is enclosed, targeting the F-$S1$-$R5$ segment list. This way, the router identifies the target path besides the origin and the protocol used to instantiate it. In the testbed case, the BGP agent generates a Command Line Interface-based script and sends it to the router management interface, triggering the filter match configuration and the steering policy action onto the selected path. The full flow computation and steering

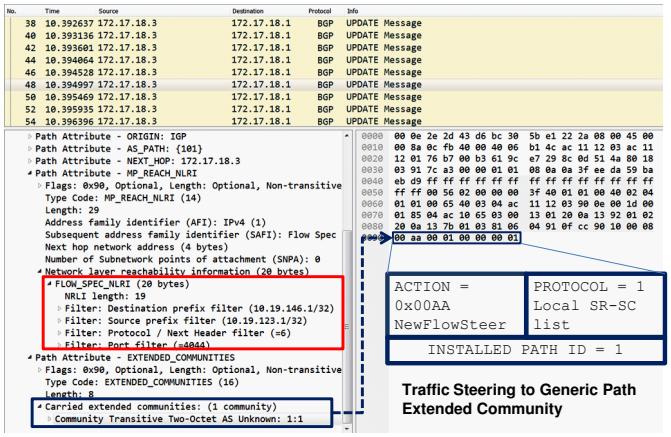

Fig. 5. Wireshark capture of extended BGP Flowspec messages enabling flow steering enforcement.

operation is performed in 2.4s. Further details are reported in Tab. I.

With reference to the configuration of Fig. 4, Fig. 6 shows a Wireshark capture collected at the controller (IP 10.30.2.73) of the OpenFlow message triggering the creation of the SR-LSP segment list at $H1$ edge node (IP 10.30.2.34). In particular, the segment list, as explained in Sec. IV-B, is created as indirect group through a `GROUP_MOD` message. The capture shows the message fields related to the creation of an indirect group, enclosing as action an array of MPLS Label Push commands (`OFPAT_PUSH_MPLS`) including the label value indication (`OFPAT_SET_FIELD`). The figure shows the details of group id 2 targeting segment list F-$S1$-$H5$ and in particular the push action details of label $H5$ (label value 1001005).

Then, Fig. 7 shows the OpenFlow steering messages generated by service chain requests computation. In particular the `FLOW_MOD` message encloses the flow match (i.e., source IP 10.19.120.2, destination IP 10.19.122.20, protocol UDP, source port 1024, destination port 1025) and the action, associating the match to the group ID 2, corresponding to the F-$S1$-$H5$ segment list. The capture shows also the `BARRIER_REQ` and `BARRIER_REP` messages at the end of the enforcement of different flows. Full computation and enforcement times, depending on the different edge node device, are reported in Tab. II.

After enforcement, flow packets sent by the traffic generator are directed to the requested service chained using the selected SR-LSPs. Fig. 8 reports the capture of packets entering (Fig. 8a) and exiting (Fig. 8b) the DPI service provided by node F running OVS. Two SC Flows are steered in the network and arrive at the service with the same MPLS stack composed of two labels (i.e., $S1$ service label 1002511, and $H5$ label 1001005). The first flow is directed to UDP port 1024, the second to UDP port 1025. After DPI inspection of the submitted packets content, the service decides to drop the second Flow. Thus, only packets belonging to the first flow are forwarded after popping the MPLS label 1002511 (label $S1$) used to require the application of the DPI service. This demonstrates the effective service chain application to such flows, without the need of any steering configuration of the service node.

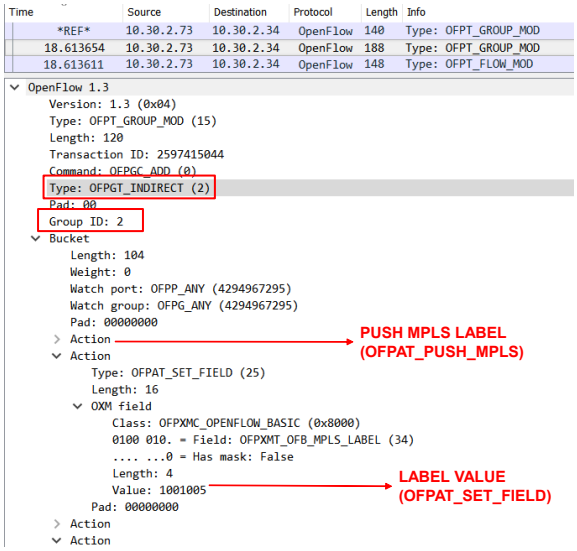

Fig. 6. Wireshark capture of OpenFlow messages setting up SR-LSP segment lists.

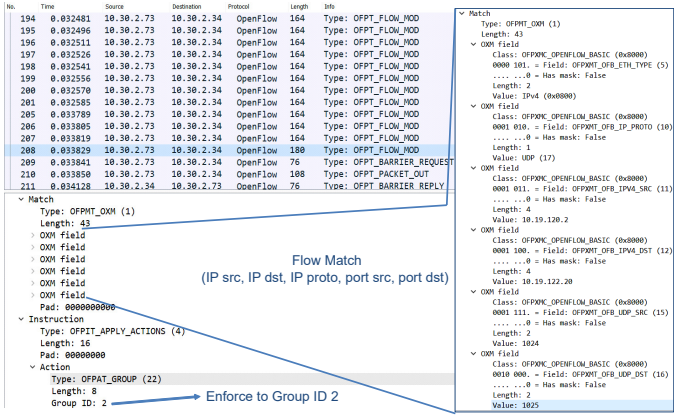

Fig. 7. Wireshark capture of OpenFlow messages enabling SDN steering.

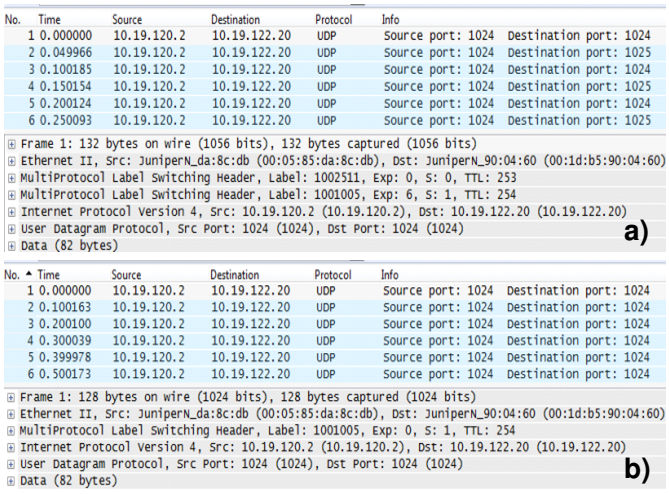

Fig. 8. Wireshark capture of flow packets enforced in SR-LSPs and steered to the DPI service: input interface (a), output interface (b) filtering packets with port 1025.

A. *Flow computation and steering scalability evaluation*

In order to assess the extended NSC Controller architecture, a scalability test evaluation has been carried out in order to analyze the performance of the flow computation algorithm described in Sec. III-B. To this end, a bulk of uniformly distributed flow requests (source, destination, bandwidth in the range 1-100 Mb/s, requested service chain) has been submitted to the Controller. The computation module has been evaluated against different network configurations. In particular, the number of installed SR-LSPs in the multi-layer network (i.e., the LSP-DB size L) supposed to reserve 10 Gb/s bandwidth each, the number of installed flows (i.e., the Flow DB size), the network dimension (expressed as the number of edge nodes e connected to a different data center) and the number of the available service chains c offered by the network are varied in order to measure the flow computation time. The reported results have been averaged on a variable number of repetitions in order to achieve the 90% of confidence interval at 10% of confidence level.

Fig. 9 reports the flow computation time as a function of the two databases, evaluated in a network with $e = 6$ edge data centers and $c = 20$ different available service chains. The behaviour is practically linear with respect to both databases size. A slight deviation occurs in the central region of each curve, e.g., in the ranges $(10^5,10^6)$ for $L = 10$ and $(10^4, 10^5)$ for $L = 10000$, where the slope increases. This is due to the fact that the number of candidate LSP paths in such ranges requires the computation of residual available bandwidth due to an increased number of flows, thus relatively delaying the flow computation. For higher values, the slope returns to initial values since the number of candidate LSPs is reduced to the increased number of installed flows fulfilling alternative paths. It has to be noted that the flow computation contribution is in the order of 1s for the extreme scaling case of 10^7 installed flows and 10^4 LSPs, whereas for reference scenarios referred to regional metro-core networks (i.e., around 10^3 LSPs and 10^5 flows) [35] the computation is below 1ms.

Fig. 10 reports the flow computation time dependence on the number of different available service chains. Reported results show that the time needed in the case of increased number of available service chain decreases, with a behavior similar to the one observed in Fig. 9 concerning the number of installed flows. The decreased time depends on the fact that the flow requests are generated with a uniform traffic matrix. Therefore, at a given size of the flow database, if the number of offered chains is higher a reduced number of flows and of LSPs offering a given service chain. As a consequence, flow computation processing operates on a reduced number of entries and decreases as shown in Fig. 9. Moreover, the service chain information is already stored in the LSP-DB, thus the selection of suitable candidates is performed with a single round of queries to the LSP-DB, not depending on the number of different available chains. This means that the offered service chain size has no practical scaling impact on flow computation. Similar considerations are applicable to the number of available edge nodes, i.e., number of client data centers attached to the SR metro-core network. In this case, the preliminary RIB query allows to identify source and destination edge nodes, and subsequent candidate LSP selection is performed with a single round of queries to the LSP-DB, being source and destination nodes specific fields of the LSP-DB entries. As a consequence, both service chains

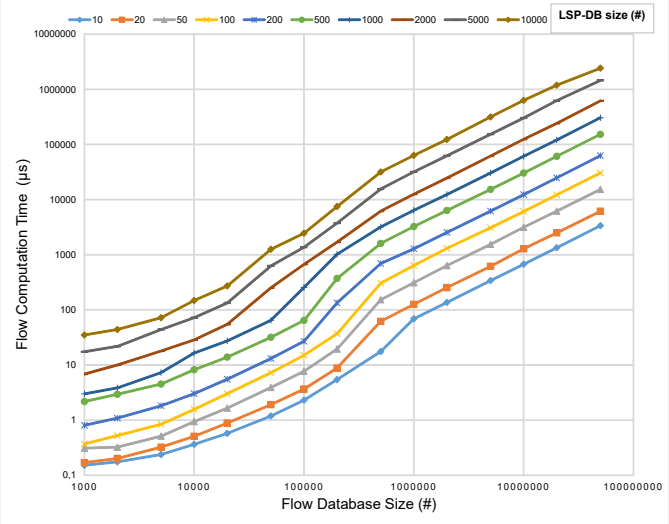

Fig. 9. Flow computation algorithm: computation time as a function of the size of active flows and active LSP in the network (e=6, c=20).

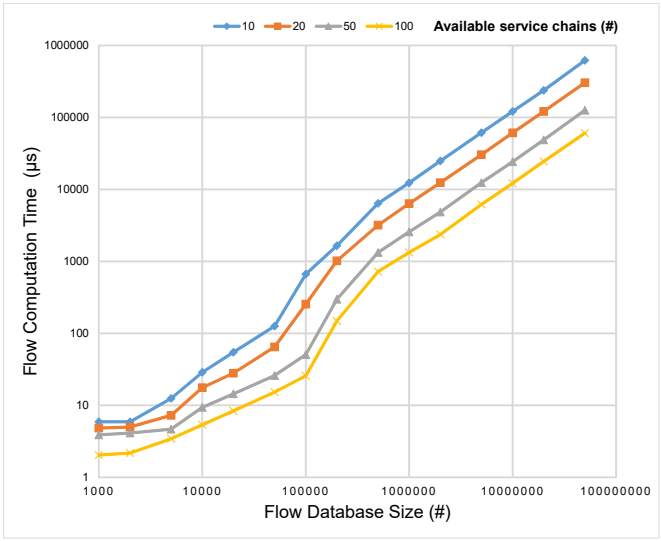

Fig. 10. Flow computation algorithm: computation time as a function of the size of active flows and number of available service chains in the network (e=6, L=1000).

and edge nodes variables have a constant impact in terms of flow computation, requiring a single query round to the LSP-DB and, most important, no direct interaction with the Flow Database.

In conclusion, the flow computation time strictly depends on the size of the stateful databases (i.e., LSP-DB and Flow DB) and it is practically independent from the number of offered service chains and the number of edge nodes. However, for database sizes comparable to those in operating metro-core networks, the flow computation time is in the order of few milliseconds, with significant scaling margins.

Besides flow computation at FCE, also flow steering scalability has been evaluated in both the aforementioned experimental testbeds. Scalability tests against the implemented BGP Flowspec API have been performed by sending bulks of different size of Flow steering messages. Results are collected by measuring at the BGP Agent (see Fig. 3) the time required to process the BGP FlowSpec messages and

TABLE I
BGP-FLOWSPEC STEERING TIME SCALABILITY

Number of Flows	BGP processing time	Overall steering time
1	144us	2.43s
10	4.1ms	2.71s
100	45ms	5.37s

TABLE II
OPENFLOW-BASED STEERING TIME SCALABILITY

Number of Flows	OVS Kernel Mode	HP 3800	HP 3500
1	0.48ms	1.48ms	3.1ms
10	0.61ms	59ms	73ms
100	20.9ms	255ms	295ms

enforce flow steering in the data plane. Results are reported in Tab. I. The second column shows the time needed to process the BGP Flowspec updates, including extensions. The third column shows the global steering time including flow source router configuration. Results show that the BGP Flowspec processing time is almost linearly increasing with the number of updates (1 update requires less than 150us, 100 Flows are processed in 45ms). The total configuration requires up to 5.4s for 100 Flows, mainly due to the router commit time. Considering that the only commit requires around 3 seconds, we can conclude that the overall BGP Flowspec steering enforcement requires around 25ms per flow in the case BGP Flowspec agent is directly embedded as router daemon service.

Scalability tests against the OpenFlow API have been performed by sending bulks of different size of flow steering messages. Results are collected by measuring at the southbound API of the controller the time between the first OpenFlow `FLOW_MOD` message of the bulk and the reception of the `BARRIER_REP` message notifying the overall steering activation. Thus, the measured time includes: the time to send the OpenFlow messages bulk, process and enforce flow steering in the specific SDN data plane edge node. Results, reported in Tab. II, show that an almost linear slope is achieved by the HP switches, targeting a 100 flows bulk in less than 300ms. In these cases, flow processing is performed by a dual-core embedded network processor inside the HP devices, with stable scaling behavior. Different behavior is observed for the OVS switch in kernel mode, achieving faster enforcement times due to the availability of increased CPU processing capabilities (i.e., dedicated QuadCore CPU in the Linux box) with respect to HP devices (100 flows are enforced more rapidly in around 20ms). In this case the steering is very fast with 1 and 10 flows while it considerably slows down with 100 flows. This behavior is quite typical in the multi-thread approach used by OVS, where the processing of few flows is totally parallelized but increasing the number of flows greatly above the number of CPUs re-introduce significant queuing time that may imply scalability issues in the case of bulks of more than 1000 flows.

VI. CONCLUSION

A Network Service Chaining control architecture conceived for microflow computation, placement and steering in a metro-core multi-layer network exploiting Segment Routing was proposed and discussed. The architecture allows the deployment of network function middleboxes in the metro-core network and service chain enforcement and steering for

microflows requiring specific QoS or security treatment. The architecture included the proposal of novel flow computation element and two alternative steering southbound API exploiting extended BGP Flowspec and OpenFlow, suitable for legacy MPLS networks and pure SDN environment, respectively. Experimental results on a SR-over elastic optical network demonstrated the effectiveness of the proposal by evaluating the scalability of flow computation in the range of few milliseconds for reference metro-core networks, the scalability of the APIs implementing the proposed extensions capable to enforce bulks of hundred microflow entries instancies within few hundreds milliseconds range. With respect to techniques currently proposed in the literature, both proposed solutions improve service chain enforcement since they do not require any control plane configuration in intermediate and destination nodes for service activation, including NSC middleboxes nodes.

REFERENCES

[1] F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi, "Service chaining in multi-layer networks using segment routing and extended BGP FlowSpec," in *Optical Fiber Communication Conference*. OSA, 2017, p. W4J.3.

[2] E. Datsika, A. Antonopoulos, N. Zorba, and C. Verikoukis, "Software defined network service chaining for OTT service providers in 5G networks," *IEEE Communications Magazine*, vol. 55, no. 11, pp. 124–131, Nov. 2017.

[3] F. van Lingen, M. Yannuzzi, A. Jain, R. Irons-Mclean, O. Lluch, D. Carrera, J. L. Perez, A. Gutierrez, D. Montero, J. Marti, R. Maso, and a. J. P. Rodriguez, "The unavoidable convergence of NFV, 5G, and fog: A model-driven approach to bridge cloud and edge," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 28–35, 2017.

[4] C. C. Byers, "Architectural imperatives for fog computing: Use cases, requirements, and architectural techniques for fog-enabled iot networks," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 14–20, 2017.

[5] N. Herbaut and N. Negru, "A model for collaborative blockchain-based video delivery relying on advanced network services chains," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 70–76, 2017.

[6] P. K. Sharma, S. Singh, Y. S. Jeong, and J. H. Park, "Distblocknet: A distributed blockchains-based secure sdn architecture for iot networks," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 78–85, 2017.

[7] Y. J. Chen, L. C. Wang, F. Y. Lin, and B. S. P. Lin, "Deterministic quality of service guarantee for dynamic service chaining in software defined networking," *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 991–1002, Dec 2017.

[8] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, "Research directions in network service chaining," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Nov 2013.

[9] R. Chaudhary, N. Kumar, and S. Zeadally, "Network service chaining in fog and cloud computing for the 5G environment: Data management and security challenges," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 114–122, Nov. 2017.

[10] A. Sgambelluri, F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi, "Experimental demonstration of segment routing," *Lightwave Technology, Journal of*, vol. 34, no. 1, pp. 205–212, 2016.

[11] A. Giorgetti, A. Sgambelluri, F. Paolucci, F. Cugini, and P. Castoldi, "Segment routing for effective recovery and multi-domain traffic engineering," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 9, no. 2, pp. A223–A232, Feb 2017.

[12] P. Marques, N. Sheth, R. Raszuk, B. Greene, J. Mauch, and D. McPherson, "Dissemination of Flow Specification Rules," *IETF, RFC 5575*, Aug. 2009.

[13] A. S. Sendi, Y. Jarraya, M. Pourzandi, and M. Cheriet, "Efficient provisioning of security service function chaining using network security defense patterns," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[14] J. Liu, Y. Li, Y. Zhang, L. Su, and D. Jin, "Improve service chaining performance with optimized middlebox placement," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 560–573, July 2017.

[15] A. A. Mohammed, M. Gharbaoui, B. Martini, F. Paganelli, and P. Castoldi, "Sdn controller for network-aware adaptive orchestration in dynamic service chaining," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, June 2016, pp. 126–130.

[16] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for nfv chaining in packet/optical datacenters," *Journal of Lightwave Technology*, vol. 33, no. 8, pp. 1565–1570, April 2015.

[17] ——, "Optical service chaining for network function virtualization," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 152–158, April 2015.

[18] R. Munoz, R. Vilalta, R. Casellas, R. Martinez, T. Szyrkowiec, A. Autenrieth, V. López, and D. López, "Integrated SDN/NFV management and orchestration architecture for dynamic deployment of virtual SDN control instances for virtual tenant networks [invited]," *J. Opt. Commun. Netw.*, vol. 7, no. 11, pp. B62–B70, Nov 2015.

[19] R. Casellas, R. Vilalta, R. Martínez, and R. Munoz, "Highly available sdn control of flexi-grid networks with network function virtualization-enabled replication," *J. Opt. Commun. Netw.*, vol. 9, no. 2, pp. A207–A215, Feb 2017.

[20] R. Martínez, A. Mayoral, R. Vilalta, R. Casellas, R. Munoz, S. Pachnicke, T. Szyrkowiec, and A. Autenrieth, "Integrated sdn/nfv orchestration for the dynamic deployment of mobile virtual backhaul networks over a multilayer (packet/optical) aggregation infrastructure," *J. Opt. Commun. Netw.*, vol. 9, no. 2, pp. A135–A142, Feb 2017.

[21] F. Lazzeri, G. Bruno, J. Nijhof, A. Giorgetti, and P. Castoldi, "Efficient label encoding in segment-routing enabled optical networks," in *2015 International Conference on Optical Network Design and Modeling (ONDM)*, May 2015, pp. 34–38.

[22] B. Raeisi and A. Giorgetti, "Software-based fast failure recovery in load balanced SDN-based datacenter networks," in *2016 6th International Conference on Information Communication and Management (ICICM)*, Oct 2016, pp. 95–99.

[23] F. Paolucci, V. Uceda, A. Sgambelluri, F. Cugini, O. G. de Dios, V. Lopez, L. M. Contreras, P. Monti, P. Iovanna, F. Ubaldi, T. Pepe, and P. Castoldi, "Interoperable multi-domain delay-aware provisioning using segment routing monitoring and BGP-LS advertisement," in *ECOC 2016; 42nd European Conference on Optical Communication*, Sept 2016, pp. 1–3.

[24] Y. Zhang, N. Beheshti, L. Beliveau, G. Lefebvre, R. Manghirmalani, R. Mishra, R. Patneyt, M. Shirazipour, R. Subrahmaniam, C. Truchan, and M. Tatipamula, "StEERING: A software-defined networking for inline service chaining," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*, Oct 2013, pp. 1–10.

[25] P. B. Pawar and K. Kataoka, "Segmented proactive flow rule injection for service chaining using sdn," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, June 2016, pp. 38–42.

[26] F. Paolucci, F. Cugini, A. Giorgetti, N. Sambo, and P. Castoldi, "A survey on the path computation element (PCE) architecture," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 4, pp. 1819–1841, Fourth 2013.

[27] F. Paolucci, A. Castro, F. Fresi, M. Imran, A. Giorgetti, B. Bhownik, G. Berrettini, G. Meloni, F. Cugini, L. Velasco, L. Potì, and P. Castoldi, "Active PCE demonstration performing elastic operations and hitless defragmentation in flexible grid optical networks," *Photonic Network Communications*, vol. 29, no. 1, pp. 57–66, Feb. 2015.

[28] F. Paolucci, A. Castro, F. Cugini, L. Velasco, and P. Castoldi, "Multipath restoration and bitrate squeezing in SDN-based elastic optical networks [invited]," *Photonic Network Communications, Springer*, vol. 28, no. 1, pp. 45–57, 2014.

[29] M. Dallaglio, N. Sambo, F. Cugini, and P. Castoldi, "Control and management of transponders with netconf and yang," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 9, no. 3, pp. B43–B52, March 2017.

[30] N. Sambo, G. Meloni, F. Paolucci, F. Cugini, M. Secondini, F. Fresi, L. Potì, and P. Castoldi, "Programmable transponder, code and differentiated filter configuration in elastic optical networks," *J. Lightwave Technol.*, vol. 32, no. 11, pp. 2079–2086, Jun 2014.

[31] R. Martinez, F. Cugini, R. Casellas, F. Paolucci, R. Vilalta, P. Castoldi, and R. Munoz, "Control plane solutions for sliceable bandwidth transceiver configuration in flexi-grid optical networks," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 126–135, August 2016.
[32] O. G. de Dios, R. Casellas, R. Morro, F. Paolucci, V. López, R. Martínez, R. Munoz, R. Vilalta, and P. Castoldi, "Multipartner demonstration of BGP-LS-enabled multidomain EON control and instantiation with H-PCE [invited]," *J. Opt. Commun. Netw.*, vol. 7, no. 12, pp. B153–B162, Dec 2015.
[33] F. Paolucci, F. Cugini, G. Cecchetti, and P. Castoldi, "Open network database for application-based control in multilayer networks," *Journal of Lightwave Technology*, vol. 35, no. 9, pp. 1469–1476, May 2017.
[34] "Open vSwitch," www.openvswitch.org.
[35] "Strongest deliverable," http://www.ict-strongest.eu/upload-files/deliverables-2/d2-1-15/download.