Segment Routing: A Comprehensive Survey of Research Activities, Standardization Efforts, and Implementation Results

Pier Luigi Ventre, Stefano Salsano, *Senior Member, IEEE*, Marco Polverini,
Antonio Cianfrani, *Member, IEEE*, Ahmed Abdelsalam, Clarence Filsfils, Pablo Camarillo, and Francois Clad

Abstract—Fixed and mobile telecom operators, enterprise network operators and cloud providers strive to face the challenging demands coming from the evolution of IP networks (e.g., huge bandwidth requirements, integration of billions of devices and millions of services in the cloud). Proposed in the early 2010s, Segment Routing (SR) architecture helps face these challenging demands, and it is currently being adopted and deployed. SR architecture is based on the concept of source routing and has interesting scalability properties, as it dramatically reduces the amount of state information to be configured in the core nodes to support complex services. SR architecture was first implemented with the MPLS dataplane and then, quite recently, with the IPv6 dataplane (SRv6). IPv6 SR architecture (SRv6) has been extended from the simple steering of packets across nodes to a general *network programming* approach, making it very suitable for use cases such as Service Function Chaining and Network Function Virtualization. In this article, we present a tutorial and a comprehensive survey on SR technology, analyzing standardization efforts, patents, research activities and implementation results. We start with an introduction on the motivations for Segment Routing and an overview of its evolution and standardization. Then, we provide a tutorial on Segment Routing technology, with a focus on the novel SRv6 solution. We discuss the standardization efforts and the patents providing details on the most important documents and mentioning other ongoing activities. We then thoroughly analyze research activities according to a taxonomy. We have identified 8 main categories during our analysis of the current state of play: Monitoring, Traffic Engineering, Failure Recovery, Centrally Controlled Architectures, Path Encoding, Network Programming, Performance Evaluation and Miscellaneous. We report the current status of SR deployments in production networks and of SR implementations (including several open source projects). Finally, we report our experience from this survey work and we identify a set of future research directions related to Segment Routing.

Index Terms—Segment routing, MPLS, IPv6, SR-MPLS, SRv6, source routing, monitoring, traffic engineering, failure recovery, path encoding, networking programming, performance, Linux, VPP, data plane, control plane, southbound APIs, northbound APIs, open source, software defined networking, SDN, service function chaining, SFC, standards.

Pier Luigi Ventre is with the Department of Electronic Engineering, University of Rome Tor Vergata, 00100 Rome, Italy (e-mail: pier.luigi.ventre@uniroma2.it).

Stefano Salsano is with the Department of Electronic Engineering, University of Rome Tor Vergata, 00100 Rome, Italy, and also with the Research Unit Roma Tor Vergata, Consorzio Nazionale Interuniversitario per le Telecomunicazioni, Rome, Italy (e-mail: stefano.salsano@uniroma2.it).

Marco Polverini and Antonio Cianfrani are with the Department of Information Engineering, Electronics and Telecommunications, University of Rome Sapienza, 00184 Rome, Italy (e-mail: marco.polverini@uniroma1.it; antonio.cianfrani@uniroma1.it).

Ahmed Abdelsalam is with the Computer Science Department, Gran Sasso Science Institute, 67100 L'Aquila, Italy, and also with the Segment Routing Architecture Group, Cisco Systems, 00142 Rome, Italy (e-mail: ahmed.abdelsalam@gssi.it).

Clarence Filsfils is with the Segment Routing Architecture Group, Cisco Systems, 1831 Brussels, Belgium (e-mail: cfilsfil@cisco.com).

Pablo Camarillo is with the Segment Routing Architecture Group, Cisco Systems, 28108 Madrid, Spain (e-mail: pcamaril@cisco.com).

Francois Clad is with the Segment Routing Architecture Group, Cisco Systems, 92130 Paris, France (e-mail: fclad@cisco.com).

Digital Object Identifier 10.1109/COMST.2020.3036826

I. INTRODUCTION

SEGMENT Routing (SR) is based on the loose Source Routing concept. A node can include an ordered list of instructions in the packet headers. These instructions steer the forwarding and the processing of the packet along its path in the network.

The single instructions are called *segments*, a sequence of instructions can be referred to as a *segment list* or as an *SR Policy*. Each segment can enforce a topological requirement (e.g., pass through a node) or a service requirement (e.g., execute an operation on the packet). The term *segment* refers to the fact that a network path towards a destination can be split into segments by adding intermediate way-points. The segment list can be included by the original source of the packet or by an intermediate node. When the segment list is inserted by an intermediate node, it can be removed by another node along the path of the packet, supporting the concept of *tunneling* through an *SR domain* from an *ingress* node to an *egress* node.

The implementation of the Segment Routing Architecture requires a data plane which is able to carry the segment lists in the packet headers and to properly process them. Control plane operations complement the data plane functionality, allowing the allocation of segments (i.e., associating a segment identifier to a specific instruction in a node) and the distribution of the segment identifiers within an SR domain.

As for the data plane, two instances of SR Architecture have been designed and implemented, SR over MPLS (SR-MPLS) and SR over IPv6 (SRv6). SR-MPLS requires no change to

the MPLS forwarding plane, see RFC 8660 [1]. SRv6 is based on a new type of IPv6 routing header called SR Header (SRH) defined in RFC 8754 [2].[1]

Temporally, SR-MPLS has been the first instance of SR Architecture, while the recent interest and developments are focusing on SRv6. In particular, the IPv6 data plane for SR is being extended to support the so-called SRv6 Network Programming Model [3]. According to this model, Segment Routing functions can be combined to achieve an end-to-end (or edge-to-edge) networking objective that can be arbitrarily complex. This is appealing for implementing complex services such as Service Function Chaining. SRv6 can be used as an *overlay* tunneling mechanism directly exposed and used by servers and as an *underlay* transport mechanism in network backbones (supporting Traffic Engineering and Resilience services). In light of this, SRv6 can simplify network architectures avoiding the use of different protocol layers.

As for the SR Control Plane operations, they can be based on a distributed, centralized or hybrid architecture. In the distributed approach, the routing protocols are used to signal the allocation of segments, and the nodes take independent decisions to bind packets to the segment lists. In the centralized approach, an SR controller allocates the segments, takes the decision on which packets need to be associated to which SR Policy and configures the nodes accordingly. Very often, an hybrid approach which consists of the combination of the previous strategies is used (see for example [4]).

The goal of this article is to provide a comprehensive survey on Segment Routing technology, including all the achieved results and the ongoing work. Hereafter (Section I-A), we start with the historical context behind the development of Segment Routing. In Section II, we provide an introduction to the main concepts of Segment Routing architecture. We consider both the SR-MPLS and the SRv6 data planes, but we focus more deeply on SRv6 which is currently attracting a lot of interest. In Section III, we provide a classification and a discussion of the standardization efforts together with an analysis of the most relevant patents. We provide a comprehensive review of research activities in Section IV, covering 88 scientific papers. The most relevant implementation results and the status of ongoing deployments are reported in Section V. We report in Section VI lessons learned and our experience. We highlight future research directions and open issues in Section VII. Finally, we include in Appendix A a table of the abbreviations.

A. Segment Routing Motivations and Evolution

The Source Routing approach consists of the inclusion of the route of the packet as a list of hops in the packet header. This has two variants. *Strict* Source Routing requires the specification of the full sequence of hops from the source to the destination. *Loose* source routing consists of specifying a list

of nodes that represent *way points* to be crossed (in their order) before reaching the destination.

These two variants of Source Routing have been considered among the possible solutions for packet routing and forwarding since the early phases of the design of packet switching technologies. In particular, they have been considered in the original definition of IPv4 protocol in the late 1970s. RFC 791 [5], which defined IPv4 in 1981, included the *Strict Source and Record Route* (SSRR) and the *Loose Source and Record Route* (LSRR) options in the IPv4 packet header. These options have been rarely used in IPv4 networks, also due to security issues. Packets carrying the SSRR or LSRR options are typically filtered (dropped) by IPv4 routers on the Internet.

Segment Routing follows the *loose* variant of Source Routing, using the same approach of IPv4 Loose Source Routing, but it is specifically based on MPLS or IPv6 data planes.

The research and standardization activities on Segment Routing originated in the late 2000s. Their main goal was to overcome some scalability issues and limitations [6], which had been identified in the traffic engineered Multi Protocol Label Switching (MPLS-TE) solutions used for IP backbones. In particular it was observed that MPLS-TE requires explicit state to be maintained at all hops along an MPLS path; this may lead to scalability problems in the control-plane and in the data-plane. Moreover, the per-connection traffic steering model of MPLS-TE does not easily exploit the load balancing offered by Equal Cost MultiPath (ECMP) routing, typically used in IP networks.

Therefore, the first important motivation and advantage of Segment Routing consists in the dramatic reduction of the *per-flow* state that needs to be maintained in network nodes to support traffic engineered paths. With Segment Routing it is not needed to configure forwarding tables in the nodes along the path, only the ingress node will store the association between a flow and its path to be enforced (i.e., the Segment List). This also means that no configuration messages (e.g., using the RSVP-TE protocol) need to be sent to all nodes along the path to establish a tunnel. The second important motivation is that using segments instead of "pinning" the full sequence of hops as with traditional MPLS-TE it is possible to exploit multiple paths (ECMP routing) within each segment.

In the early 2010s, the IETF started the "Source Packet Routing in Networking" Working Group (SPRING WG) to deal with Segment Routing. The activity of the SPRING WG has included the identification of Use Cases and Requirements for Segment Routing (for example, [7], [8], and [9] have become IETF RFCs). In July 2018, the SPRING WG has issued the "Segment Routing Architecture" document (RFC 8402 [10]) along with another informational RFC on monitoring use cases [11]. More recently (December 2019) other 3 RFCs have been issued, while many other documents are still under discussion by the WG, as it will be analyzed later in this article.

Looking at the scientific bibliography, the seminal paper on the Segment Routing Architecture is [12]. Published in 2014, it provides an overview of the motivations for SR, describes a set of important use cases and illustrates the architecture.

---

[1]Such a header has not been specified so far for IPv4; thus, segment routing is not possible in current IPv4 networks. Of course, it is possible to encapsulate IPv4 packets *inside* SR-MPLS and SRv6 packets. In fact, SR-MPLS and SRv6 are commonly used as transport networks for IPv4 traffic.
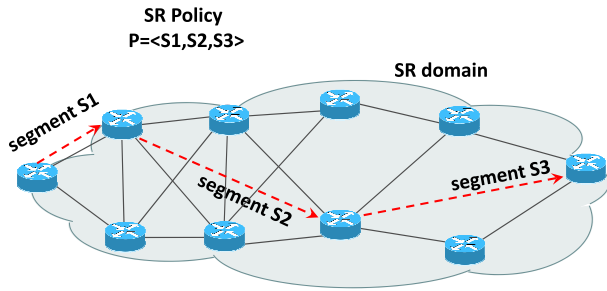
Fig. 1.   SR policy and segments.

Fig. 2.   Segment Routing operations.

The basic concepts proposed in [12] have been elaborated and refined in the RFC 8402 [10].

Currently, Segment Routing is receiving a lot of interest from operators for its applications in different types of networks (transport backbones, access networks, datacenters and 5G networks).

The MPLS based data plane (SR-MPLS) relies on the well established MPLS technology. SR-MPLS can be seen as an improvement and a simplification of the traditional MPLS control plane, so it is beneficial to operators with an already deployed MPLS infrastructure. The IPv6 based data plane (SRv6) is gaining traction as it offers the possibility to combine overlay and underlay networking services and features only using the IPv6 technology. The SRv6 *network programming model* offers unprecedented flexibility in designing and operating network services, so SRv6 is an attractive choice for operators that are deploying new networks or planning the evolution of their networking architectures.

We conclude this short historical review by noting that a very large number of patents (about 900) have been registered related to Segment Routing, as it is possible to verify with a cursory on-line search. We provide an overview and analysis on the most relevant patents in Section III, they further demonstrate the high interest of vendors and service providers in SR technology.

II. THE SEGMENT ROUTING (SR) ARCHITECTURE

This section includes a short tutorial on the main aspects of SR architecture. Our goal is to provide a common ground and a conceptual framework reference for the survey. RFC 8402 [10] represents the most important source of information for SR architecture. The work in [13] (published in 2017) provides a short and effective introduction to Segment Routing architecture, with a focus on the MPLS data plane. The survey paper [14] has a section about SR architecture, which attempts to give more details related to both data plane (for SR-MPLS) and control plane aspects.

Following the RFC 8402, let us start by discussing the general concepts of SR, which are independent from the specific data plane (MPLS or IPv6). A simple example of an SR path composed of three *segments* (S1,S2,S3) is shown in Fig. 1. We can refer to the list of segments as an *SR Policy*: the Segment Routing policy P = <S1,S2,S3> consists of steering the packets through node S1, then through node S2 and then to the destination S3. The ordered list of segments (segment list) is
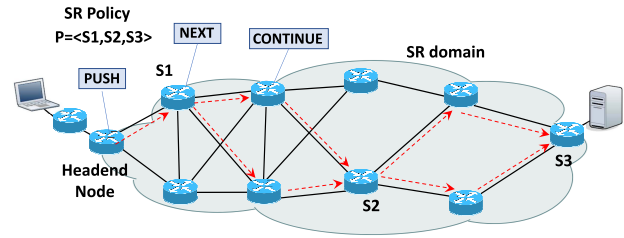
inserted in the packet headers by the source node of the SR Policy. The Segment Routing domain (*SR domain*) is the set of nodes participating in the source-based routing model.

A segment is described by a Segment Identifier (*Segment ID* or *SID*). For the MPLS data plane, a SID is an MPLS label, while for the IPv6 data plane a SID is an IPv6 address. As shown in Fig. 2, the segment list is added to the packet headers by a *headend* node that *steers* the packets of a flow onto the SR policy. The headend node can be the originator of the packet (e.g., an host) or an intermediate node that performs a classification of the traffic and associates the SR policies to the packets (as in Fig. 2). In other words, the hosts *can* be part of an SR domain, but this is not required and depends on the overall scenario in which SR is applied. It is expected that all nodes in an SR domain are managed by the same administrative entity. For example, a Service Provider backbone can constitute an SR domain and the headend node will be the ingress edge router of the backbone (in this case, the hosts are not part of the SR domain).

Three basic operations on SIDs and segment lists have been defined for a generic SR data plane: PUSH, NEXT and CONTINUE. In the examples of Fig. 1 and Fig. 2, We assume for simplicity that S1 and S2 represent topological instructions and S3 is the destination node of the SR policy P, so that the policy P instructs the packet to cross two nodes identified by the SIDs S1 and S2 (in this order) and then to reach the node identified by the SID S3.

The PUSH operation consists of the insertion of a segment on top of the segment list, i.e., as the new first segment of the SR policy. In order to build the SR policy P described above, the headend node executes the PUSH operations in this order: PUSH(S3), PUSH(S2), PUSH(S1). In an SR packet, the segment that specifies the instruction to be executed is called the *active* segment. In the considered example with the SR policy P, the headend node will send the packet with active segment S1.

The NEXT operation is executed by a node that has processed the active segment and considers the next segment of the SR policy to be executed. In our example, the node identified with SID S1 receives the packet and performs the NEXT operation. The next segment is S2, which becomes the active segment so that the packet is forwarded toward S2. The NEXT operation also covers the case of the last node of an SR policy, in which the NEXT operation usually results in processing the packet according to regular IP forwarding.

Finally, the CONTINUE operation is performed by nodes that are in the path between two segments. For example, the

intermediate nodes in the path between S1 and S2 perform the CONTINUE operation. The path between S1 and S2 is not prescribed by the SR policy and will be chosen considering the regular IP routing toward node S2 in the SR domain. If there are multiple equal cost paths between nodes S1 and S2 (as in Fig. 2) and the ECMP (Equal Cost MultiPath) mechanism is supported by the IP routing in the SR domain, it can be conveniently exploited by Segment Routing.

The segments can be classified into *Global Segments* and *Local Segments*. Global Segments correspond to instructions that are globally valid in an SR domain. Local segments correspond to instructions that are valid within a single node.

The typical example of a global segment is an instruction to forward packets toward a given destination IP network or a destination IP node. Considering that an IGP (Interior Gateway Protocol) routing protocol (e.g., OSPF or ISIS) is used in the SR domain, these instructions are called *IGP-prefix segment* and *IGP-node segment* (or simply prefix segment and node segment). All nodes in the SR domain can execute the prefix segment or node segment instructions by considering the path toward the destination network or destination node in their routing table.

The most important example of local segment is the instruction to forward a packet to a node identified as adjacent by the IGP routing protocol. This corresponds to sending the packet on a specific outgoing interface and can be executed only by a specific node. This instruction is called *IGP-Adjacency Segment*. Thanks to the use of IGP-Adjacency segments, it is possible to prove that any path across an SR domain can be expressed by an SR Policy (which can include a combination of global and local segments) [15]. Local segments can also be used to represent service instructions to be executed in a given node. The mapping of global and local segments into Segment Identifiers (SIDs) and the distribution of the SIDs in an SR domain are different for SR-MPLS and SRv6 and will be discussed in the next subsections.

The *IGP-Anycast Segment* is an IGP-Prefix segment that corresponds to an anycast prefix, i.e., a prefix advertised by a set of routers that can be used for High Availability or Load Balancing purposes.

The *Binding Segment* is used to associate an SR policy (i.e., a Segment List) to a SID (called Binding SID or *BSID*) in a given node. This means that the node that processes the Binding SID replaces this segment with a Segment List: a packet received with the BSID as active segment will be steered according to the associated SR policy. In this way, the packet classification can be executed by a node X that adds the Binding SID in the SR Header. The node X does not need to know the details of the SR policy to be applied (i.e., the Segment List). Thanks to the BSID, the packet will be forwarded to a node Y, which is able to apply the SR Policy.

As an example, the node X can classify traffic for a given destination network N that requires "low latency" and traffic for the same destination network N that requires "low loss". Node Y is an ingress node of a backbone that provides connectivity toward the network N. Two SR policies ("low latency" and "low loss") are used to forward traffic toward the network N across the backbone. The respective lists of segments can

TABLE I
MAPPING SR CONCEPTS INTO SR-MPLS AND SRv6

Generic SR	SR-MPLS	SRv6
SR Policy	Label Stack	Segment List (of IPv6 addresses) in the SR Header
Active Segment	Topmost Label	IPv6 address indicated in the IPv6 Destination Address
PUSH Operation	Label Push	Adding an IPv6 in the Segment List in the SR Header
NEXT Operation	Label POP	Decrementing the Segment Left field, copying the active segment in the IPv6 Destination Address
CONTINUE Operation	Label Swap	Forwarding according to IPv6 Destination Address

change over time, based on Traffic Engineering considerations. Upon these changes, the node Y is re-configured to apply the current SR Policy to the packets identified by the Binding SID. Node X does not need to be reconfigured, as the Binding SIDs remain constant over time. This approach improves the scalability, resilience and service independence of the solutions based on Segment Routing.

Table I summarizes the mapping of the SR concepts into the two data planes (MPLS and IPv6) and will be discussed in the next two subsections.

It is interesting to note how some requirements that led to the definition of the SR solutions are currently fulfilled by "Over the top" (OTT) providers to deliver services with a lower degree of flexibility, using tunneling technologies such as GRE (Generic Routing Encapsulation) [16] and VXLAN (Virtual eXtensible LAN) [17]. These technologies allow to encapsulate traffic and forward packets toward remote nodes according to an overlay logical topology. Unfortunately, they come with a penalty, for example a protocol like VXLAN needs an L3 underlay to transport traffic and loses full L3 forwarding capabilities such as ECMP forwarding [18]. They are not really forms of source routing and do not allow the user to define way-points to which traffic can be directed. To further elaborate, in [18] it is reported an interesting use case where multi-tenancy in a datacenter fabric has been implemented using SRv6 as overlay/underlay instead of the commonly used technologies such as VXLAN with a drastic simplification of the architecture.

As of Service Function Chaining (SFC) scenarios, the Network Service Header (NSH) [19] is a solution that works on top of the tunneling technologies. Therefore, NSH can be used in conjunction with Segment Routing, when SR is used only as a tunneling mechanism (enhanced with Traffic Engineering features). On the other hand, NSH can be seen as an alternative to Segment Routing for implementing the Service Function Chaining functionality. In this respect, the works in [20] and [21] elaborate on Service Function Chaining scenarios where SRv6 would allow for a full replacement of the NSH layer, leading to a simplification of the infrastructure and a reduction in the burden on the devices.

A. MPLS Data Plane (SR-MPLS)

The MPLS data plane (SR-MPLS) is specified in [1]. For SR-MPLS, Segment Routing does not require any change to

the MPLS forwarding plane. An SR Policy is instantiated through the MPLS Label Stack: the Segment IDs (SIDs) of a Segment List are inserted as MPLS Labels. The classical forwarding functions available for MPLS networks allow implementing the SR operations. The PUSH operation corresponds to the Label Push function, i.e., pushing an MPLS label on a packet. The NEXT operation corresponds to the Label Pop function, i.e., removing the topmost label. The CONTINUE operation corresponds to the Label Swap function, i.e., associating an incoming label with an outgoing interface and outgoing label and forwarding the packet on the outgoing interface. The encapsulation of an IP packet into an SR-MPLS packet is performed at the edge of an SR-MPLS domain, reusing the MPLS Forwarding Equivalent Class (FEC) concept. A Forwarding Equivalent Class (FEC) can be associated with an SR Policy.

The mapping of Segments to MPLS Labels (SIDs) is a critical process in the SR-MPLS data plane. In general cases, different routers in the SR domain could have different available ranges of labels to be used for Segment Routing. Therefore, each router can advertise its own available label space to be used for Global Segments called *SRGB - Segment Routing Global Block* (in general, this label space can even be composed of a set of non-contiguous blocks). For this reason, in the SR domain Global Segments are identified by an index, which has to be re-mapped into a label, taking into account the node that will process the label. This methodology has been also covered by the patent [22] and defines the following.

Assuming that the SRGB of a node is a label range starting from 10000, for a Global Segment with index X, the node needs to receive the label 10000+X. As an example, in Fig. 3 A we consider how to implement the SR policy described in Fig. 2 using the SR-MPLS data plane. We assume that different nodes are using different SRGBs. The SRGBs of the nodes and the segment index associated to the segments S1, S2 and S3 are shown in the gray rectangle. The headend node needs to consider in advance which is the SRGB of the nodes that will perform the NEXT operation to the segments, because the label for the next segments needs to be crafted accordingly. In particular, the initial label for segment S2 set by the headend node will be 1002, i.e., the SRGB of node S1 (1000) plus the index for segment S2 (2). Node S1 will have to modify the label to 4002 if the packet is forwarded to node N4 (whose SRGB is 4000) or to label 6002 if the packet is forwarded to node N6 (whose SRGB is 6000). Both nodes N4 and N6 will remap (swap) the label to 2002 when forwarding the packets to S2. The initial label for node S3 set by the headend node is 2003, i.e., the SRGB of node S2 (2000) plus the index for segment S3 (3). This label will reach node S2 unmodified, then it will be properly processed by node S2 that will remap (swap) it considering the SRGB of the next hop in the path toward node S3.

This remapping process complicates operations and troubleshooting. There are also services (e.g., involving anycast segments) that cannot be realized if different SRGBs are used by different nodes. For this reason, it is strongly recommended in [10] that an identical range of labels (SRGB) is used in all routers, so that a Global Segment will always be mapped to the
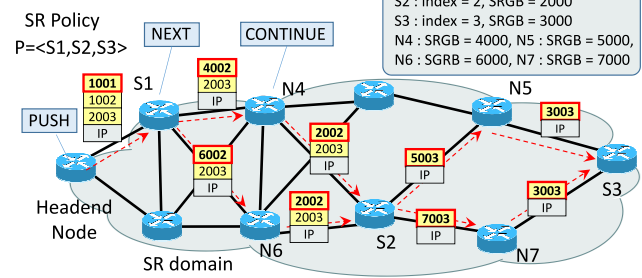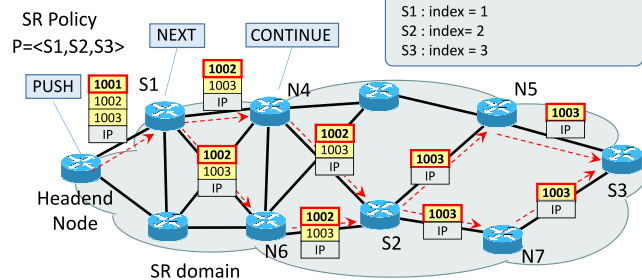

Fig. 3.   SR-MPLS data plane: mapping segments to labels using the SRGB.
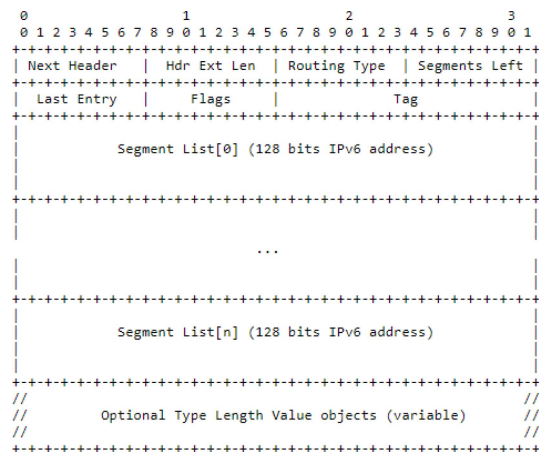
Fig. 4.   Segment Routing Header.

same SID (MPLS label) in all nodes. In Fig. 3 B we present the mapping of the same SR policy under the suggested operating mode in which an identical SRGB is used in all nodes. We observe that the MPLS labels do not need to be remapped, so that the same label consistently identifies the same global segment throughout the SR domain.

B. IPv6 Data Plane (SRv6)

For the IPv6 data plane (SRv6), a new type of IPv6 Routing Extension Header, called Segment Routing Header (SRH) has been defined in [2]. The format of the SRH is shown in Fig. 4. The SRH contains the Segment List (SR Policy) as an ordered list of IPv6 addresses: each address in the list is a SID. A dedicated field, referred to as *Segments Left*, is used to maintain the pointer to the active SID of the Segment List.

In order to explain the SRv6 data plane, we consider three categories of nodes: Source SR nodes, Transit nodes and SR

Fig. 5.   SRv6 data plane operations.
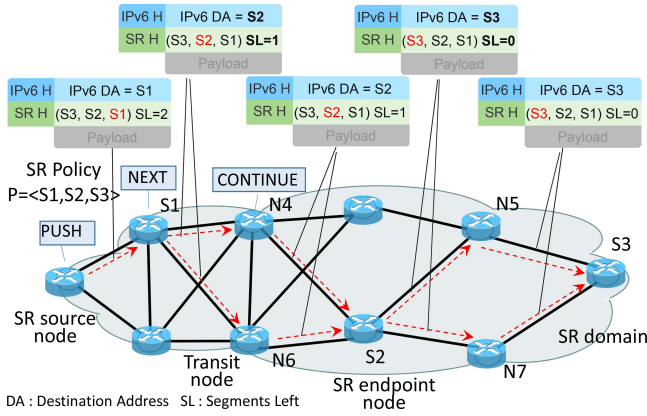
Trasport Layer Header, TCP or UDP), without encapsulating the packet in a new IPv6 packet. This option only applies to IPv6 packets and it is especially suited in case the source host is acting as Source SR node (Headend node).

In addition to the basic operations (PUSH/ NEXT/ CONTINUE), the *SRv6 Network Programming* model [3] describes a set of functions that can be associated to segments and executed in a given SRv6 node. Examples of such functions are: different types of packet encapsulation (e.g., IPv6 in IPv6, IPv4 in IPv6, Ethernet in IPv6), corresponding decapsulation, lookup operation on a specific routing table (e.g., to support VPNs). The list of functions described in [3] (discussed in Section II-C) is not meant to be exhaustive, as any function can be associated to a segment identifier in a node. Obviously, the definition of a standardized set of segment routing functions facilitates the deployment of SR domains with interoperable equipment from multiple vendors.

According to [3], we can revisit the notion of Segment IDentifier (SID) taking into account that IPv6 addresses are used as SIDs in SRv6. A 128 bit SID can be logically split into three fields and interpreted as LOCATOR:FUNCTION:ARGS (in short LOC:FUNCT:ARG) where LOC includes the L most significant bits, FUNCT the following F bits and ARG the remaining A bits, where $128 = L + F + A$.

The LOC corresponds to an IPv6 prefix (for example with a length of 48, 56 or 64 bits) that can be distributed by the routing protocols and provides the reachability of a node that hosts a number of functions. The length L of the locator is not fixed and can be chosen by each operator for its own SR domain (also independently for different nodes). All the different functions residing in a node can share the same locator and have a different FUNCT code, so that their SIDs will be different. From the routing point of view, the solution is very scalable, as a single prefix is distributed for a node, which implements a potentially large number of functions, with limited impact on the routing tables of the nodes in the SR domain. The ARG bits can be used to provide information (arguments) to a function. They are optional: if $A = 0$, the SID can be simply decomposed in two fields as LOC:FUNCT, and $128 = L + F$.

A SID, that splits into LOC:FUNCT:ARG, is a global segment if the LOC prefix is routable in the SRv6 domain, which is the typical case. It is also possible to define local segments in SRv6, i.e., non routable SIDs that can be executed by a node and need to be preceded by a global SID used to forward the packet to the node. In any case, the use of such local SIDs can be avoided as the FUNCT part of a global SID in the form LOC:FUNCT:ARG can represent the needed local function.

C. SRv6 Network Programming Model

The SRv6 Network Programming model is defined in [3]. It consists of combining functions that can reside in different nodes to achieve *a networking objective that goes beyond mere packet routing*. The functions described in [3] can support valuable services and features such as layer 3 and layer 2 VPNs, traffic engineering and fast rerouting. The Network Programming model offers the possibility to implement virtually any service by combining the basic

Segment Endpoint nodes. A Source SR node corresponds to the *headend* node discussed above. It can be a host originating an IPv6 packet, or an SR domain ingress router encapsulating a received packet. In Fig. 5 we consider the latter case, the Source SR node is an edge router that encapsulates a packet (which can be IPv6, IPv4 or even a Layer 2 frame) into an outer IPv6 packet and inserts the SR Header (SRH) as a Routing Extension Header in the outer IPv6 header. The encapsulated packet is indicated as Payload in Fig. 5. The Segment List in the SRH is composed of S1, S2 and S3 which are stored in reverse order (the fist SID is S3, the last segment in the SR policy). The Segment Left field is set to 2, so that the active segment is S1, represented in red in the figure. The Source SR node sets the first SID of the SR Policy (S1) as IPv6 Destination Address of the packet. These operations correspond to a sequence of the PUSH operations described above.

The SR Segment Endpoint node receives packets whose IPv6 destination address is locally configured as a segment. The SR Segment Endpoint node inspects the SR header: it detects the new active segment, i.e., the next segment in the Segment List, modifies the IPv6 destination address of the outer IPv6 header and forwards the packet on the basis of the IPv6 forwarding table. These operations correspond to the NEXT operation described above. In Fig. 5, we can see that S1 is the first SR Endpoint node, it decrements the Segment Left fields to 1, making S2 the active segment, and sets S2 as IPv6 destination address.

A Transit node forwards the packet containing the SR header as a normal IPv6 packet, i.e., on the basis of the (outer) IPv6 destination address, because the IPv6 destination address does not locally match with a segment. These operations correspond to the CONTINUE operation. In Fig. 5, nodes N4, N5, N6 and N7 are Transit nodes, which perform a regular forwarding of the packet toward the IPv6 destination address. Note that in SRv6 the Transit nodes do not need to be SRv6 aware, as every IPv6 router can act as an SRv6 Transit node.

In the given example, the PUSH operation is performed by encapsulating a packet (IPv6, IPv4 or Layer 2 frame) into an outer IPv6 packet with a Segment Routing Header. Another possibility is to perform the *insertion* of an SRH as a new header between the IPv6 header and the Next Header (e.g., the

functions in a *network program* that is embedded in the packet header.

As shown in Fig. 4, the SRH can include an optional section that carries Type Length Value (TLV) objects. These TLV objects can be defined to transport information that needs to be elaborated by one or more segments of an SR policy (Segment List). For example, the so-called HMAC TLV can be added and used to verify that an SRH header has been created by an authorized node and that the segment list is not modified in transit. Another potential use of TLV objects is for exchanging Operation and Maintenance (OAM) information among the nodes of the SR domain.

The draft [3] defines two different sets of SRv6 behaviors, known as *SR policy headend* and *endpoint* behaviors. With reference to Fig. 5, *SR policy headend* behaviors are executed in the SR source nodes, while *endpoint behaviors* in SR endpoint nodes (e.g., S1, S2, S3). *SR policy headend* behaviors steer received packets into the SRv6 policy, matching the packet attributes. Each SRv6 policy has a list of SIDs to be attached to the matched packets. Note that in an earlier version of [3], the *SR policy headend behaviors* were referred to as *transit* behaviors, which was misleading because the same attribute (*transit*) was applied to the SR source nodes and to the transit nodes not fulfilling any operation. On the other hand, an SRv6 *endpoint* behavior, also known as behavior associated with a SID, represents a function to be executed on packets at a specific location in the network. Such function can be a simple routing instruction, but also any advanced network function (e.g., firewall, NAT).

Table II reports a non-exhaustive list of SRv6 behaviors, listing the documents that provide their description. The *H.Encaps* behavior encapsulates an IPv6 packet, which becomes the inner packet of an IPv6-in-IPv6 packet. The outer IPv6 header carries the SRH header, which includes the SIDs list. The *H.Encaps.L2* behavior is the same as the *H.Encaps* behavior, with the difference that *H.Encaps.L2* encapsulates the full received layer-2 frame rather than the IP packet (Ethernet over IPv6 encapsulation). The *H.Insert* behavior inserts an SRH in the original IPv6 packet immediately after the IPv6 header and before the transport level header. The original IPv6 header is modified; specifically, the IPv6 destination address is replaced with the IPv6 address of the first segment in the segment list, while the original IPv6 destination address is carried in the SRH header as the last SID of the SIDs list.

The *End* behavior represents the most basic SRv6 function among the endpoint behaviors. It replaces the IPv6 destination address of the packet with the next SID in the SIDs list. It then forwards the packet by performing a lookup of the updated IPv6 destination address in the routing table of the node. We will refer to the lookup in the routing table as *FIB lookup*, where FIB stands for Forwarding Information Base. The *End.T* behavior is a variant of the *End* behavior, in which the FIB lookup is performed in a specific IPv6 table associated with the SID rather than in the main routing table. The *End.X* behavior is another variant of the *End* behavior, in which the packet is directly forwarded to a specified layer-3 adjacency bound to the SID, without performing a FIB lookup of the IPv6 destination address.

TABLE II
(NON-EXHAUSTIVE) LIST OF SRv6 BEHAVIORS

Behavior	Defined in
H.Encaps	srv6-network-programming [3]
H.Insert	srv6-network-programming [3]
H.Encaps.L2	srv6-network-programming [3]
End	srv6-network-programming [3]
End.T	srv6-network-programming [3]
End.X	srv6-network-programming [3]
End.DT4	srv6-network-programming [3]
End.DT6	srv6-network-programming [3]
End.DX4	srv6-network-programming [3]
End.DX6	srv6-network-programming [3]
End.DX2	srv6-network-programming [3]
End.AS	service-programming (SFC) [23]
End.AD	service-programming (SFC) [23]
End.AM	service-programming (SFC) [23]
T.M.Tmap	mobile-uplane [24]
End.M.GTP4.E	mobile-uplane [24]
End.M.GTP6.D	mobile-uplane [24]
End.M.GTP6.E	mobile-uplane [24]

The *End.DT6* behavior pops out the SRv6 encapsulation and performs a FIB lookup of the IPv6 destination address of the exposed inner packet in a specific IPv6 table associated with the SID. It is possible to associate the default IPv6 routing table with the SID, in this case the inner IPv6 packets will be decapsulated and then forwarded on the basis of its IPv6 destination address according to the default routing of the node. The *End.DX6* behavior removes the SRv6 encapsulation from the packet and forwards the resulting IPv6 packet to a specific layer-3 adjacency associated to the SID. *End.DT4* and *End.DX4* are respectively the IPv4 variant of *End.DT6* and *End.DX6*, i.e., they are used when the encapsulated packet is an IPv4 packet. The *End.DX2* behavior is used for packets encapsulated at Layer 2 (e.g., with H.Encaps.L2). It pops out the SRv6 encapsulation and forwards the resulting L2 frame via an output interface associated to the SID.

D. Control Plane for SR and Relation With SDN

Control Plane operations are needed to complement the data plane functionality and provide a complete solution for Segment Routing. The Control Plane can be based on a fully distributed approach, in which the routers have the capability to take independent decisions to setup and enforce SR Policies. It can rely on a centralized SR controller that takes a decision and instructs the routers according to SDN principles, or on a combination of the two approaches (hybrid solution).

For the SR-MPLS data plane, the definition of a fully distributed approach has been worked out within the IETF, with the definition of extensions to the IGP routing protocols (OSPF, ISIS, see [25], [26], [27]). These extensions to the routing protocols are used by each router to advertise the different types of IGP-segments (prefix, node, adjacency, anycast) and to distribute some SR configuration data. All other routers in the SR domain will receive this information by means of the IGP routing protocol. This information is needed to map the segments included in an SR policy into SIDs represented as MPLS labels in the SR-MPLS data plane. As we have discussed in Section II-A, in the general case each router

could allocate different ranges of labels to be used for Global Segments. The range of labels used for the global segments by a router, called *SRGB - Segment Routing Global Block* is among the SR configuration information advertised using the routing protocol. We recall that it is strongly recommended to use an identical range of labels (SRGB) in all routers.

For the IPv6 data plane, the process of advertising the IGP-prefix, IGP-node and IGP-anycast segments is simplified thanks to the use of IPv6 addresses as SIDs. In particular, there is no need to extend the IGP routing protocols to distribute these segment types, as they are natively distributed by the routing protocols. Also the definition of a Segment Routing Global Block as in the SR-MPLS is not needed, and the operations related to Global Segments can rely on IPv6 addresses that are globally routable in the SR domain. This means that the Control Plane for SRv6 can use the regular IGP routing protocols (OSPF, ISIS) to support the basic operations, while extensions are still needed ([28], [29]) to distribute IGP-Adjacency segments and other SR configuration information.

The definition of the control plane for Segment Routing has started from the SR-MPLS data plane and then the SRv6 data plane has inherited most of the functionality, which has been adapted to the new data plane. We observe that an original design goal of the control plane for Segment Routing has been to support the fully distributed approach, in which routers are capable of taking autonomous decisions. This allows the same functionality of a traditional MPLS network to be offered, which does not need a centralized SDN controller for its operations. On the other hand, we observe now a trend to focus on an hybrid approach in which distributed routing protocols coexist with an SR controller. This hybrid approach is aligned with the vision of Software Defined Networking, which aims to remove complexity from the devices and to centralize control plane function in SDN controllers.

In light of this, the Segment Routing architecture can be deployed by seeking the right balance between distributed and centralized control. The distributed control is used by the routers to exchange reachability information and evaluate the shortest paths in a traditional way; with no need to interact with the centralized controller. We observe that this is the best approach to provide connectivity in Wide Area Networks in which the control connections between the nodes and the SDN controllers are affected by non-negligible latency and failure probability. Segment Routing can be still used for Fast Reroute by pre-configuring SR policies that provide alternate paths in case of link or node failures, and these are automatically activated by the node when the failure happens.

The pre-calculation of such SR policies can be performed in a distributed mode or can be centralized in a controller; these concepts have also been explored in [30], which patents a method for the orchestration of SR based WANs. Basic topology information and additional information for Traffic Engineering need be conveyed to the controller, as well as service related information that is advertised by nodes using distributed routing protocols. The SDN controller can receive this information in different ways. For example, it can participate to the IGP routing protocol or it can interact with routers in a proprietary way to extract their IGP databases. Otherwise, it can receive information by routers using extensions to BGP-LS (BGP-Link State) [31]. Whatever mechanism has been used to retrieve the needed information from the nodes, the SDN controller is in charge of taking decisions about the SR policies that implement advanced features or services such as Traffic Engineering, VPNs or Service Function Chaining. This approach allows the clear decoupling of the data plane operations from the service logic operating in the control plane.

The mechanisms and protocols for the SDN controller to enforce the SR policies by configuring the nodes are left open in the definition of SR architecture. As mentioned in [10], some options are Network Configuration Protocol (NETCONF), Path Computation Element Communication Protocol (PCEP) [32], and BGP. The OpenFlow protocol can be used as a mechanism to configure the SR policies only for SR-MPLS, while the processing of the SRv6 header is not supported by the latest standard version of the protocol. An Open Source implementation of a SouthBound API for SRv6 based on gRPC is reported in [4]. The main characteristic of the Segment Routing solution compared to other SDN solutions is that only the edge nodes need to be configured to enforce a given SR policy, while the internal nodes do not need to keep state per SR policy. This feature gives a substantial improvement in terms of scalability.

E. Segment Routing Motivations and Use Cases

As anticipated in the introduction section, the RFC 5439 [6] has identified some scalability issues of traditional MPLS networks with Traffic Engineering support. These issues gave rise to an interest in defining a more scalable solution such as Segment Routing back in the late '00s. Several use cases and requirements for Segment Routing have been collected in a number of documents.

In [7], the main use cases identified are: MPLS tunneling (i.e., to support VPN services), Fast ReRoute (FRR) and Traffic Engineering (further classified in a number of more specific use cases). A set of Resiliency use cases is described in [8]. In [9], the Segment Routing use cases for IPv6 networks are considered with a set of exemplary deployment environments for SRv6: Small Office, Access Network, Data Center, Content Delivery Networks and Core Networks.

III. STANDARDIZATION ACTIVITIES AND PATENTS

In this section we propose a classification and description of the standardization activity related to Segment Routing and the analysis of the most relevant patents. We have classified 14 Request For Comment (RFC) and 50 Internet Drafts. Our taxonomy is based on 7 categories and the result of the classification is shown in Table III.

Hereafter we discuss the categories of the classification and then in the next subsection we report an overview of the key standardization activities. The first category is *Architecture*, where all the documents regarding the description of the general architecture of a Segment Routing network are considered. The RFC 8402 [10] falls into this category and describes the

TABLE III
CLASSIFICATION OF THE STANDARDIZATION ACTIVITIES DOCUMENTS

Architecture			[1], [3], [10], [23], [33]–[36]
Use-case and Requirements			[7]–[9], [24], [37]–[44]
Fast Reroute (FRR)			[45]–[47]
OAM			[11], [48], [49], [49]–[51]
Performance Measurement			[52]–[56]
Protocol Extensions	Data Plane	SR-MPLS	[57]–[61]
		SRv6	[2], [62], [63]
	Control Plane	BGP	[64]–[66]
		BGP-LS	[67]–[74]
		IS-IS	[27], [28], [75]–[79]
		OSPF	[25], [26], [29], [80] [75], [76], [81]
		PCEP	[32], [82]
		LISP	[83]

main features of SR, such as the source routing paradigm idea, the concept of SID and the definition of the two supported data planes. In the category *Use-case and Requirements* the documents describing use case scenarios for SR, e.g., use of SR in WANs, data center networks, mobility and network slicing, are inserted. Specifically, in this category there are three RFCs: i) RFC 7855 [7] introducing the Source Packet Routing in Networking (SPRING), ii) RFC 8355 [8] related to network resiliency using SR, and iii) RFC 8354 [9] that describes how to steer IPv6 or MPLS packets over the SPRING architecture.

The third category is *FRR* one, i.e., Fast Reroute realized through SR. The main standardization activity in this category is related to fast recovery after a link failure, and is referred to as Topology Independent Loop Free Alternate (TI-LFA), described in [45]. No RFC has been published in this category. *Operations, Administration, and Maintenance* (OAM) is the fourth defined category, where we include all the standardization activities related to tools used for maintenance of the network. As example, RFC 8287 [48] focuses on the implementation of the ping and traceroute tools in SR-MPLS, while in [49] the same is done for SRv6. In the *Performance Measurement* category we consider all the documents describing measurement procedures related to performance parameters, such as delay and packet loss, in an SR network. We include in this category also the two specifications RFC 6374 [55] and RFC 7876 [56] that explain how to measure delay and packet loss in MPLS. Despite these two documents have not been produced during the standardization activities of SR, we decided to include them in Table III since they are massively used in the drafts for performance monitoring regarding SR. Finally, the *Protocol Extensions* category covers two different set of documents related to extensions of legacy protocols: i) data plane protocols extensions, and ii) control plane protocols extensions. As for the data plane, we include RFC 8660 describing SR-MPLS [1] and RFC 8754 describing SRv6 [2]. As for the control plane, we the consider the documents on modifications to routing protocol (e.g., BGP and OSPF) for the dynamic distribution of the SIDs in the SR network, or control protocol for the communication between a central controller (in case of centralized control plane) and the devices at the data plane (e.g., PCEP).

As regards the patents, we have reported 18 documents. Our analysis has covered different type of documents including use case driven patents. Section III-B elaborates on the surveyed documents. Among these, there are 4 patents covering the founding principles of Segment Routing: [84], [85], [86], and [87]. Another group of patents are related to the extensions of the routing protocols that are needed to transport SR related configurations [31], [88], and [89]. Then, there is a large group of documents covering specific mechanisms and use cases: [22], [30], [90], [91], [92], [93], [94], and [95]. Finally, during our research we have found a number of recent documents like [96], [97], and [98] that show the centrality of Segment Routing also in the deployment of future networks.

A. Key Standardization Efforts

In this subsection, we provide an overview of the most important standardization efforts, by considering 9 documents among the almost 70 listed in Table III. The works [10] and [33] define key tenets of the SR architecture and discuss the benefits brought by SR in terms of scalability, privacy and security. The support of key use cases like NFV/SFC, SD-WAN and next generation of mobile networks are evaluated in [23], [40], and [24]. Extension of basic SR concepts [3] and SR-based Fast Re-route (FRR) mechanisms against single failures [45] are also proposed. Finally, the improvements of the routing stability and extensions to the routing protocols are analyzed in [46] and [75].

The Segment Routing architecture and its overall design is described in [10]. It defines the concept of a segment as a network instruction and presents the basic types of segments: prefix-SID, adjacency-SID, peering-SID and binding-SID. It also explains how such segments can be attached to data packets, leveraging the MPLS or IPv6 data planes, in order to steer traffic flows on any path in the network without requiring any per-flow state in the fabric.

The concept of an SR policy is detailed in [33]. It explains how Candidate Paths are defined as explicit SID-lists or as dynamically computed paths based on some optimization criteria, and how the active Candidate Path is selected. Moreover, it presents various ways of steering traffic into an SR Policy, automatically by coloring BGP service routes, remotely using a Binding-SID, or statically with route policies. The concepts described in this draft equally apply to the MPLS and SRv6 data planes.

The SR architecture is extended from the simple steering of packets across nodes to a general network programming approach in [3]. Using this framework, it is possible to encode arbitrary instructions and not only locations in a SID-list. Each SID is associated with a function to be executed at a specific location in the network. A set of basic functions are defined in [3], but other functions can be defined by network operators to fit their particular needs. Moreover, SID arguments allow functions to be provided additional context or their behavior to be tweaked on a per-flow basis.

Reference [23] defines the service SIDs and describes how to implement service programming (i.e., Service Function

Chaining) in SR-MPLS and SRv6 enabled networks. The key tenet is to associate a SID to each network function (either physical or virtual). These service SIDs may be combined together in a SID-list and finely programmed by leveraging the network programming concept. They can also be combined with other types of SIDs to provide traffic engineering or VPN services. Service segments can be associated to legacy appliances (*SR-unaware* VNFs, i.e., VNFs with no SRv6 capabilities), thanks to the SR proxy mechanisms which perform the SR processing and hide the SR information from the VNF. The three *endpoint* behaviors that has been defined in [23] for supporting Service Function Chaining are: *End.AD*, *End.AS* and *End.AM*. The first two implement respectively a *static* and a *dynamic* SRv6 proxy for SR-unaware Virtual Network Function (VNF). They support IPv6 SR packets in *H.Encaps* mode. The SRv6 proxy intercepts SR packets before being handed to the SR-unaware VNF, hence it can remove the SR encapsulation from packets. For packets coming back from the SR-unaware VNF, the SR proxy can restore the SRv6 encapsulation updating the SRH properly. The difference between the *static* and the *dynamic* proxies is that the SR information that needs to be pushed back in the packets is statically configured in the first case and it is *learned* from the incoming packets in the *dynamic* case. Instead, *End.AM* implements the *masquerading* proxy that supports SR packets travelling in *H.Insert* mode.

Enabling underlay Service Level Agreements (SLA) for a VPN in a scalable and secure way using SR, while ensuring service opacity, is explained in [40]. SR based VPNs are analyzed considering the case of a single provider and of multiple providers. Moreover, the draft addresses the control plane aspects of such solution, which are managed by an over the top SD-WAN controller. Finally, the benefits brought by the SR technology to VPN services are analyzed in term of scalability, privacy, billing management and security.

The applicability of SRv6 to the user plane of mobile networks is described in [24]. Three modes are addressed: traditional mode, enhanced mode and enhanced mode with interworking. In the traditional mode, the mobile user plane is unchanged except for the use of SRv6 as transport instead of GTP-U [99]. Enhanced mode uses only SRv6 and its programming framework. Finally, the enhanced mode with interworking uses SRv6 but provides also interworking functionality with legacy components still using GTP. The document describes a mechanism for end-to-end network slicing and defines the SRv6 behaviors for the SRv6 mobile user plane. Among these behaviors, the most important ones define the functions for the coexistence of GTP-U [99] and SRv6. In particular, *T.M.Tmap* translates a GTP-U over IPv4 packet to a SRv6 packet. Its counterpart is *End.M.GTP4.E*, which maps an SRv6 packet to a GTP-U over IPv4 packet. Finally, *End.M.GTP6.D* and *End.M.GTP6.E* define respectively the translation of GTP-U over IPv6 packet to a SRv6 packet and SRv6 packet to a GTP-U over IPv6 packet.

Topology Independent Loop-free Alternate (TI-LFA) [45] provides Fast Re-Route (FRR) mechanisms protecting against link, node or local Shared Risk Link Groups (SRLGs) failures in SR enabled networks. For each destination in the network, a backup path is pre-computed and installed in the forwarding table, so that it is ready to be activated as soon as a failure is detected. For each destination, the backup path matches the post-convergence path, which is followed by the traffic after the network convergence. The draft analyzes also the benefits of using Segment Routing with respect to traditional FRR solutions.

A mechanism leveraging SR to prevent transient routing inconsistencies during the convergence period that follows a network topology modification, is described in [46]. Instead of directly converging to a new next-hop after a topology modification, a node can temporarily steer the impacted traffic through a set of loop-free SR Policies, thus preventing it from being affected by routing inconsistencies. After the network has fully converged, the temporary SR Policies are removed with no impact on the traffic.

A set of extensions to the IGP routing protocols that enable Prefix-SIDs to be associated with operator-defined shortest path algorithms, called SR Flexible Algorithms (Flex-Algo), are introduced in [75]. These algorithms are defined as an optimization metric (IGP, TE or delay) and a set of constraints (e.g., resources to be excluded from the path). Each node participating in a Flex-Algo computes the shortest paths to the Prefix-SIDs of that Flex-Algo and installs them in it forwarding table. SR Flexible Algorithms allow traffic to be steered along traffic-engineered paths such as low-latency or dual-plane disjoint path with a single Prefix-SID.

B. Relevant Patents

In this section we reviewed 18 among almost 1000 patents that can be found in patent search engines. It is interesting to note that SR is the central technology of the patent only in around 200 of the 1000 patents. For most of the patents there is a corresponding standardization activity (e.g., an Internet draft or an RFC) promoted by the same vendor that produced the patent.

We will start by analyzing [84], [85], [86], and [87] which cover key tenets of the SR architecture.

Despite the fact that the name refers to MPLS and SR, a primitive architecture for Segment Routing is patent in [86]. It has likely been one of the first patents related to Segment Routing and basically describes a mechanism to map egress ports of the routers to labels, to advertise these data with a link state protocol and to forward data packets in the network with this information attached.

The patent in [84] is related to RFC8402 [10], and it includes the definition of a SID list and explains how a node can build such a stack of labels; it shows the forwarding behavior of the SR nodes and makes a clear distinction between the operations of an IP router, Label/MPLS router and SR router. Moreover, the patent covers the different types of SIDs, and for each them a definition is provided. Finally, different use cases for SR are illustrated.

Another fundamental document [85], which shares a lot of concepts with [2], patents the usage of an IPv6 extension header for native implementation of Segment Routing in IP networks. The document also describes the SRv6 Extension

Header along with the operations performed by the nodes to update the extension header and IPv6 header upon the reception of a packet. Instead, the applicability of Segment Routing in a scenario where a computer network has been partitioned in multiple sub-networks is described in [87].

The second group of documents mainly describe extensions to the routing protocols. For example, extensions for BGP-LS to signal SIDs within an IGP domain are described in [31]. A method for the creation and maintenance of SR policies through BGP protocol is proposed in [88]. The method relies on receiving BGP update messages with additional metadata referring to an SR policy. Several scenarios regarding the advertisement of adjacency segment identifiers are covered in [89]. The method applies both to point-to-point links and LAN segments.

The vast majority of patents elaborate on the support of key use cases such as TE, SD-WAN, network monitoring, NFV/SDN, SFC and so on. Different methods and an architecture for the monitoring of network paths which leverage SR technology are described in [90]. According to the patent, nodes can assemble a SID list with the aim of testing/monitoring a specific network path. It is possible find a lot of similarities with the works of Section IV-A.

The use of SR in a Label Distribution Protocol (LDP) domain and the coexistence of MPLS nodes with nodes that are SR enabled is faced in [91]. The patent also includes the definition of hybrid nodes that are capable of operating both as LDP enabled nodes and SR aware routers.

Three patents are related to fast reroute mechanisms and loop avoidance. In [92], a method to detect and avoid loops into SR traffic engineered paths is described. It is based on the idea of defining for each tunnel a second path, which is exploited when the traffic is received from a node internal to the network. Instead, various methods for loop avoidance upon network failures are defined in [93]. Finally, FRR mechanisms for Segment Routing based networks are described in [94].

A method for the orchestration of connectivity services in an SR-enabled WAN is described in [30]. The controller is considered outside the network. It is able to monitor the current state of the WAN and leverages a solver to perform traffic engineering. Decisions taken by the solver are enforced using SR technology. A method for the assignment of the SIDs, based on assigning a global index value to each node and transmitting a base value to calculate the final segment identifiers, is proposed in [22]. The patent in [95] describes an algorithm to "translate" an explicit path into a set of SIDs (like the works in Section IV-E). The main idea of the proposed algorithm is to compute, starting from the source node, the shortest path towards the farthest node without ambiguity, i.e., with no multiple equal cost paths: a SID of the resulting Segment List is associated to each farthest node.

An analysis of the most recent patents highlights that SR represents a key technology also for the deployment of future networks in many different scenarios: i) in [96], the implementation of a network slicing mechanism exploiting an SR-flexible algorithm is proposed; ii) in [97], a mechanism to report mobile node location information in 5G networks

TABLE IV
CLASSIFICATION #1 BASED ON RESEARCH CATEGORIES

Category	References
Monitoring (MON)	[101]–[108]
Traffic Engineering (TE)	[109]–[130]
Failure Recovery (FR)	[131]–[139]
Centrally Controlled Architectures (CCA)	[4], [140]–[154]
Path Encoding (PEN)	[15], [155]–[161]
Network Programming (NP)	[21], [100], [162]–[167]
Performance Evaluation (PEV)	[168]–[171]
Miscellaneous (MISC)	[14], [172]–[180]

using SR is proposed; iii) in [98], the flexibility of path encoding provided by SR is exploited to define a Virtual Machine migration procedure with zero loss [100].

IV. RESEARCH ACTIVITIES

In this section, we describe research activities related to SR, and we provide two different classifications to characterize research papers on the basis of their main scope. We also show how to extract useful information regarding ongoing SR research activity from analyzing the relationship between the two classifications proposed.

The first classification proposed is based on the identification of seven main *Research Categories*, as reported in Table IV. The first one is the *Monitoring* category, collecting all the works that describe and implement tools related to network monitoring activities. Some examples include the measurements of the end to end delay over a given input route or the assessment of the volume of the traffic flows. The second category is *Traffic Engineering*, where we include all works proposing advanced routing strategies to optimize network performances. The third category is *Failure Recovery*, covering solutions to provide fast network recovery in the case of a node/link failure. Due to the time scale constraints, the works in this category are based on local mechanisms, i.e., not involving the central controller. The fourth category defined is *Centrally Controlled Architectures*, including all papers focusing on the implementation of an SR network with a centralized control plane realized on top of an underlay network (IP, SDN, MPLS). Here we point out that, despite some works classified as *Traffic Engineering* being based on a centrally controlled architectures, they are not included in the *Centrally Controlled Architectures* category. This is due to the fact that their main scope remains to optimize a TE goal (such the reduction of the congestion, or the minimization of energy consumption). In the *Path Encoding* category, we group all the papers that propose algorithms aiming to translate network paths into an SL. Specifically, we take as input a path in the form of a sequence of nodes and links that the generic path encoding algorithm provides as output a sequence of SIDs to be pushed in the packet header in order to steer the packet along the input path. The sixth category is *Network Programming*, where we inserted scientific works proposing solutions that exploit the programmability feature of SR, i.e., using service based SIDs to define the functions to be executed on packets crossing a specific segment list. A significant example of works falling into this category include works related to Service Function
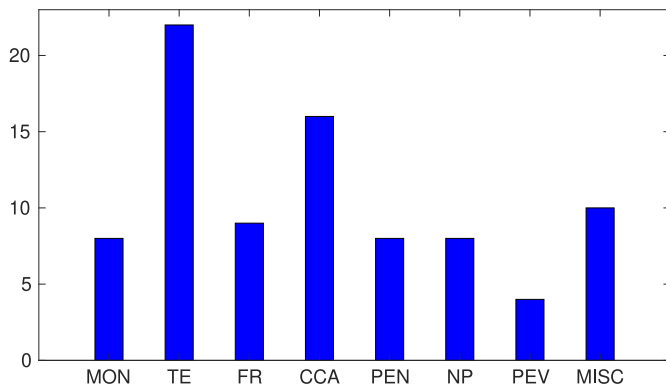
Fig. 6. Number of references for each category of the defined taxonomy.

Fig. 7. Graph showing the relation between the research categories and the SR topics.

Chaining. Finally, we define a *Miscellaneous* category, where we place all works not belonging to previous categories.

In Fig. 6 we can see an histogram showing the number of references falling into each of the defined categories. Analyzing Fig. 6, it is evident that *Traffic Engineering* and *Centrally Controlled Architectures* are the most investigated subjects, while the other categories have been covered by almost the same number of works. This behavior is quite expected since the main feature of SR is to define routing paths in a very flexible way and to make use of widely deployed data planes. This phenomenon is really appealing for the definition of new solution to TE problems and for the realization of overlay networks. On the contrary, the number of papers related to *Network Programming* could indicate a low interest in such a novel topic among the research community. We believe that *Network Programming* represents a highly interesting research topic for near future, and that the lower number of available papers is only due to its early definition state.

In order to better investigate the research activity related to SR, we propose a second classification based on the specific SR topics considered in the research works. The new SR-related classification is reported in Tab. V. The different SR topics are also aggregated into three main groups:

- SR feature exploitation;
- SR functions optimization;
- SR extensions.

The *SR feature exploitation* group covers all works making use of SR features to solve classic networking problems, such as network resource optimization and performance improvement. The first SR topic is *routing flexibility*, i.e., the possibility of steering a packet over a non trivial path (e.g., containing ECMP, loops, etc.). The second feature is the *source routing*, i.e., the capability of SR to instruct only the source node for the configuration of a specific network path. The third SR feature is *programmability*, i.e., the possibility to force a packet to go through a function by using specific SIDs. All the remaining SR features are included in the *other* topic. Here, we merged the following topics together: i) the Adj-SIDs, used to force a packet to be forwarded on a specific output port; ii) the ECMP, i.e., the ability of SR to balance the traffic over multiple paths provided by the IGP routing protocols; iii) the Type Length Value (TLV) used to add optional data to the SR header; iv) the Binding SID (BSID) that allows a user to define SR tunnels
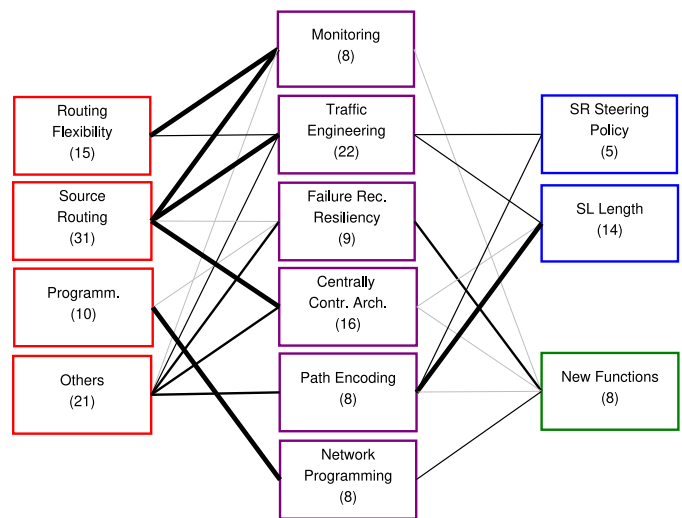
in a transit node; v) the SR traffic counters (Base Pcounters and TM Pcounters), which collect traffic statistics based on the active segment carried in the packet headers; and vi) the spray policy, which allows the duplication of an incoming packet over multiple output ports and using different SLs.

In the *SR functions optimization* group, there is a prominence of research activities aimed at improving the inner functions of an SR network. The first topic of the group is the *SR Steering Policy*, i.e., the definition of policies to be installed on network devices in order to attach the proper SL to each incoming packet. The second topic is *SL length*, i.e., the number of SIDs of a segment list, that has an impact on SR header insertion and processing.

The last group, i.e., *SR extensions*, is represented by a single topic (*new functions*) and is related to new functionality implemented in SR to support advanced services.

Tab. V reports the classification of the research works according to the SR topics. Differently than Tab. IV, where the same reference can appear only once, in Tab. V, it can be classified under different categories. As a matter of example, the *Adj-SIDs* SR feature is used in [159] in order to perform *SR functions optimization* from the point of view of *SL length*.

Tab. V shows that the most used SR feature is the *source routing* one, while there is still a limited amount of works focusing on *network programmability* and on the definition of *new functions*.

In order to obtain further insights into SR research activity, we defined a way to show the relationship between classifications reported in Tab. IV and Tab. V, respectively. In Fig. 7 we can see a graph which is defined in the following way: each node represents a research category (violet rectangles at the center of the figure) or an SR topic (divided in red, blue and green rectangles at the border of the figure), and an edge among a category and an SR topic is present only if both are present in the same work. Moreover, the thickness of an edge depends on the number of papers covering the specific category/topic pair. The graph in Fig. 7 shows several interesting outcomes: i) the works related to the *Monitoring* category mainly exploit the routing flexibility and the source

TABLE V
CLASSIFICATION #2 BASED ON SR TOPICS

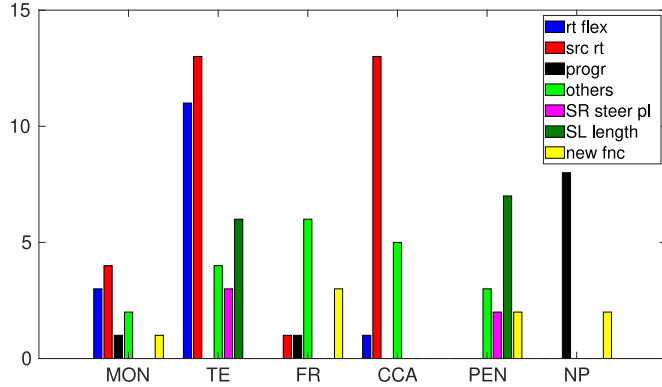| SR feature exploitation				SR functions optimization		Extend SR
Routing Flexibility	*Source Routing*	*Programmability*	*Others*	*SR steering policy*	*SL length*	*New Functions*
[102],[105],[106],[110] [112],[113],[115],[118] [119],[122],[127],[128] [129], [130], [154]	[101],[102],[103] [104],[109],[110] [111],[114],[116] [117],[118],[119] [121],[123],[124] [125],[126],[135] [140],[141],[142] [143],[144],[145] [146],[147],[148] [149],[150],[151] [152]	[21],[108],[138],[162] [163],[164],[165] [166],[167] [100]	**BSID:**[131],[139],[148],[149],[151] [154] **ECMP:**[116],[122],[123] **TLV:**[108],[131],[138],[141] **Adj-SID:**[15],[121],[156],[158],[159] **Spray:**[132],[133] **Traffic counters:**[107]	[111],[119],[121] [160],[161]	[15],[109] [114],[121] [122],[126] [127],[155] [156],[157] [158],[159] [160],[161]	[15], [108] [135],[136] [137],[161] [165],[100]

Fig. 8.   Histogram showing the relation between the Classifications defined in Tab. IV and Tab. V.

TABLE VI
CLASSIFICATION OF THE REFERENCES RELATED TO MONITORING

Objective	References
Delay measurements	[101],[108]
Health checking of network devices	[102],[103],[104]
Traffic measurements	[105],[106],[107]
Traceroute	[108]

- *traffic measurements*, aiming to assess the traffic matrix of a network exploiting routing flexibility and SR counters;
- *traceroute*, aiming to implement the well-known traceroute utility in SR.

In the following paragraphs, we provide a brief overview of the references classified as monitoring related works.

A novel monitoring system powered by Segment Routing (SR), used for the provisioning of delay-aware network services spanning multiple-domains, is proposed in [101]. Based on SR-MPLS principles, it enables delay measurements over multiple candidate routes without requiring related LSP signalling sessions. The authors consider two types of probes using SR-MPLS. The first type is originated and terminated by network stations, allowing the retrieval of round-trip measurements only. They also have less accuracy. Moreover, they are typically used for performance measurements over a single link. Instead, the second type relies on external monitoring components, which inject and receive timestamped probes routed according to the enforced SR segment list. The second type of measurement requires synchronization between the end-points, but also allows for the measurement of unidirectional delay, which is more useful when it is necessary to deploy services in the network. Along with most of related works, the project shares the objective of the reduction of control plane complexity through SR. However, from the paper is not clear which implementation the authors used for SR, or if they relied on any vendor solution.

The traffic steering capabilities of SR have been used in SCMon [102], a new solution for continuous monitoring of the data-plane. It allows the user to track the health of all routers and links: i) forcing "monitoring probes to travel over cycles"; and (ii) testing "parallel links and bundles at a per-link granularity". The key insight is that network nodes compute a second network graph and calculate routes on this monitoring topology, which spans all network links. Nodes then carefully select ECMP paths and enforce packet forwarding through

routing paradigm; ii) the same result is obtained for *Traffic Engineering* works, but in this case SR steering policy and SL length topics are also covered; iii) in order to provide *Failure Recovery* solutions, new functions generally need to be defined; iv) source routing is the most used SR feature for *Centrally Controlled Architecture* solutions, since it allows the reduction of communication between the central controller and network devices; v) as expected, *Path Encoding* works are mainly focused on the optimization of the SL length (it is in any case interesting to note that the Adj-SID is the main SR feature used to achieve this scope); vi) finally, works falling into the *Network Programming* category always exploit the programmability feature of SR, and, in some cases, new functions need to be defined.

In Fig. 7, we report for each category/SR topic the number of related references.

In the next subsections we briefly describe all research works, considering the classification proposed in Tab. IV.

A. Monitoring

Eight research works proposing monitoring solutions able to exploit SR have been defined, as shown in Table VI. These works have been classified on the basis of their main aim:

- *delay measurements*, aiming to obtain the delay for links and routers exploiting the possibility of modifying the SL at source nodes, i.e., source routing SR feature;
- *health checking of network devices*, aiming to monitor the network state exploiting the capability of SR in defining ad-hoc routes, i.e., routing flexibility feature.

cycles leveraging SR in order to detect/localize failures and overloading of single/multiple links. A prototype implementation of SCMon has been evaluated on publicly available topologies and emulated networks. In the first experiment, the ratio of number of cycles over number of edges is evaluated to analyze the percentage of the networks covered. The authors then evaluate the time to detect black holes, showing that most of them are detected within less than 100 msec. The work's results are very interesting and can be applied to real use cases. However, an open-source implementation is not available at the time of writing.

In [103], the authors focus on bandwidth-efficient monitoring schemes based on cycles. They propose four different algorithms to compute cycles which are designed to traverse/cover every link in the network. This optimization based on cycles allows network resources to be saved and the network to be monitored from a single point of advantage. Segment Routing is used as transport technology to forward the probes along the network. The paper builds upon the results of [102]: the authors leverage the phase 1 of SCMon to build the monitoring topology of the network, and then apply their two-phase algorithms in order to minimize the cycle cover length. Performance evaluation shows the effectiveness of these algorithms in terms of cycle cover length, segment list depth and improvements with respect to the baseline (SCMon).

Instead, three ILP formulations for the construction of monitoring-cycles are described in [104], builds on [103]. A first ILP formulation solves optimally the problem of covering every link in the network using monitoring-cycles with minimum cycle cover length. To further conserve network bandwidth, the first formulation is extended to jointly minimize the total segment list size needed. Finally, since the time required to detect a network failure is affected by the longest cycle, the first formulation is also extended to minimize the length of the longest cycle jointly.

The flexibility of SR is exploited in [105] to perform traffic measurements and obtain the Traffic Matrix. A traffic measurement is performed by rerouting a flow and checking the load variations caused on the network links. Even though the idea of measuring traffic through routing perturbations is not new, SR turns out to be an enabling technology for the applicability of such an approach. In fact, while in the past traffic flows were rerouted by acting on the OSPF link weights, causing routing instability and performance degradation, SR allows the modification of a path by simply acting on the ingress node, then reducing the impact of a rerouting. In [105], the problem of assessing the TM while minimizing the routing perturbations is formulated as an ILP and an heuristic algorithm called SERPENT is presented. Due to the high computational complexity of SERPENT, a lighter greedy heuristic called PaCoB is proposed in [106].

An attractive feature of SR is the introduction of specific interface counters that allow statistics on network traffic flows to be obtained. If this feature is included in the hardware design of the router, the update of the traffic counters can be associated with the normal processing in the SR data plane, thus having a negligible impact on the router performance. The simplest type of SR counters, named Base Pcounters,

collect traffic statistics (byte/packets) passing through a router and having a specific active segment. Enhanced SR counters, named TM (Traffic Matrix) Pcounters, allow the user to distinguish between traffic that is internal to the SR domain and flows that are injected into the SR domain. Specifically, a TM Pcounter collects traffic statistics of traffic flows received by an interface marked as external (this is a configuration option for the network operator). Since TM Pcounters can discriminate packets on the basis of the incoming interface, using TM Pcounters provides a thinner granularity with respect to using Base Pcounters and facilitates the estimation of the Traffic Matrix. Starting from the availability of this new type of traffic related information, in [107] the Traffic Matrix Assessment problem has been extended to include SR counter-measurements. The authors show that, depending on the structure of the Segment Lists used in the network, the Traffic Matrix can be assessed with no error.

In [108] the Linux kernel is extended to run eBPF programs as in-network functions attached to the SRv6 SIDs; further details about the implementation are provided in Section V. The authors demonstrate the effectiveness of their approach by building three different applications. The first one offers a passive monitoring solution of network delays (direct links or end-to-end paths) [53]. The high level idea is that a small percentage of traffic is encapsulated with a special SRH carrying on additional information like timestamps. These timestamps are then used by the recipient nodes to calculate one-way or round-trip delays. The second application realizes a link aggregation group using SRv6. In particular, a weight-round-robin scheduling is realized to aggregate the bandwidth of two different links. Finally, an enhanced version of traceroute has been realized, implementing a new SRv6 behavior, the so-called *End.OAMP*. This behavior, when triggered, performs a *fib lookup* in the node and return to a destination address specified in the SRH all the ECMP next hops. If possible, this function is leveraged at each hop, otherwise the program reverts to the legacy ICMPv6 mechanism.

In the following paragraphs we provide a comparison of the works classified in the *Monitoring* category.

Regarding the solutions related to delay measurements, both ([101] and [108]) allow the one-way end to end delay between two points in the network to be obtained. While the former requires the use of an external monitoring tool to generate timestamped probes (the use of SR is limited to the creation of the end to end path), the latter does not. In any case, the solution proposed in [108] is suitable only for networks realized by means of Linux based SRv6 routers, since it exploits the definition of eBPF programs to perform the measurement.

There are two SR based monitoring tools for checking the health status of the network links. The first is ([102], and the second is a tool proposed in [103] and [104]). The approach they follow is similar, i.e., the creation of cyclic paths through SR where probe messages are sent. With respect to [102], the tools in [103] and [104] optimize the use of the monitoring resources needed to check the status of all the network elements. In any case, link bundles are not supported in [103] and [104], differently than in [102].

TABLE VII
CLASSIFICATION OF THE REFERENCES RELATED
TO TRAFFIC ENGINEERING

Objective	*Energy Consumption*	[112],[113]
	Link Bandwidth	[109], [110],[111],[114],[115] [116],[117],[118],[119],[121] [122],[123],[124],[125],[126] [127],[128][129],[130]
	Rejected Requests	[109],[118]
Takes into account	*Delay*	[110],[118],[126],[127]
	SR impact (overhead, flow state)	[109],[111],[121],[122],[123] [124],[125],[126],[127]
Scenario	*Full SR*	[109],[111],[112],[113],[114] [115],[116],[117],[118],[120] [122],[124],[125],[126],[127] [129]
	Partially deployed SR	[110],[121],[130]

Among the Traffic measurement tools based on SR, two different approaches can be identified in [105], [106], [107]. The first approach [105], [106] aims at measuring traffic flows by causing link load variation through re-routing operations. The second approach [107] is based on the use of specific traffic counters, available in SR enabled nodes. While both approaches offer comparable performances in terms of the quality of the assessed Traffic Matrix, the counter based solution does not affect the network performance. This is different from the two re-routing based solutions.

B. Traffic Engineering

Due to its appealing features in terms of routing flexibility, SR is widely used to face Traffic Engineering related problems. During our investigation, we have found twenty two papers exploiting SR to provide advanced TE solutions. TE research works all employ the classic structure of an optimization problem: i) an objective function must be minimized/maximized, ii) taking into account a set of parameters, and iii) considering a specific network scenario. Three different TE objectives have been covered by the literature, i.e., the minimization of network energy consumption, the optimization of congestion and the minimization of the number of rejected requests. The high routing flexibility allowed by SR might cause the presence of complex and long network paths. For this reason, while optimizing routing according to a specific objective, it is important to take into account the impact that excessively complex routing solutions might have on the network performance. As well as end to end delay, some of the reviewed works also take into account SR related overhead, both in terms of bandwidth wasted due to the insertion in the packets of the SR header, and the number of SR steering policy to be configured in the edge routers. Finally, the considered network can be a full SR one, i.e., all the nodes are SR capable, or a partially deployed SR, where only a subset of nodes can process the SR header.

Table VII show the classification of the TE related references. It is interesting to note that most of the works consider the congestion minimization as the main objective, and that there are few solutions that can work also in a hybrid network scenario.

In the following paragraphs we provide a brief overview of the references classified as TE related works.

The TE algorithm based on SR for an SDN infrastructure presented in [109], is able to build a path with bandwidth guarantees and to minimize at the same time the possibility of rejecting traffic demands. With respect to other solutions, it takes into account the "link criticality" and not only link residual bandwidth. Citing [109]: "Link criticality is based on the concept of the minimal interference routing method". It minimizes the possibility of rejecting requests when the network becomes overloaded. The proposed algorithm not only achieves the goal of balancing network traffic load, but it also promises to reduce network costs. Since it is based on SR principles, the proposed solution also considers the extra network overhead caused by the segment labels in the packet headers. The path length has been modeled as a constraint of the heuristic, adopting an extra hop limitation in order to save network resources - extra bandwidth used by the segment lists in the packet headers. According to the authors, the time complexity of the algorithm can meet the requirements of dynamic online routing. However, there are no open source implementations available, and only simulation results are provided by the authors.

Bahnasse and Louhab [110] propose an SDN based architecture for managing MPLS Traffic Engineering Diffserv-aware networks, which bases its forwarding on SR-MPLS principles. This architecture is supposed also to support hybrid deployments whereby SDN equipment coexist with legacy devices, guaranteeing the same forwarding capabilities. Legacy devices are confined in the core of the network while SDN capabilities have to be supported by edge devices. In this way, once the controller has calculated the paths meeting the SLA parameters of the flows, it programs the network interacting with the edge and setting up the SR paths. Then over time the controller monitors the network and dynamically manages the SR-LSPs in order to ensure that the routing realized by segments does not violate end-to-end QoS constraints.

Segment Routing and Multicast are combined in [111]. The authors propose a routing solution for Multicast based on SR and an heuristic for Multicast tree calculation with bandwidth guarantees, which takes into account the load of the links, the number of branching points and the state in the network. The objective is to minimize the number of requests being rejected. In particular, the SDN controller computes an explicit Multicast tree using the aforementioned heuristic, and then programs the source node of the tree and its branching points: each time a packet reaches a branching point it needs to be duplicated and forwarded on different paths, and a modification of the segment lists is performed. Simulation results show that the proposed method outperforms other routing algorithms, although an open source implementation is not available and deeper analysis is not feasible.

SR based TE is also investigated in [112]. In particular, the authors design an online energy-efficient Traffic Engineering method. They use the SDN controller to selectively switch off and put in sleep mode a subset of links. They then dynamically adapt the number of the powered-on links to the traffic load. In this work, SR is used to dynamically re-route traffic. First,

a least-congested link technique is run to identify the eligible links that can be switched-off. At this point new routes are calculated, thus solving via heuristics the "Multicommodity Flow Problem". Finally, the SDN controller enforces new paths through SR or IGP forwarding is leveraged if the route corresponds to the shortest path. At this point it is not necessary to turn the links off. Similarly to other works, the problem is solved by identifying first the hop-by-hop paths and then mapping them into SR paths keeping the constraint of the fixed routing over the given hop-by-hop path. The authors of these works provide an interesting analysis implementing the solution in the OMNET++ simulator. However, only numerical results are given, and as far as we know an open source implementation is not available.

The flexibility in path selection achieved by SR is exploited in [113] to propose a new energy efficiency routing strategy. The main focus of the work is to switch off a subset of network links by properly selecting alternative paths for the traffic currently steered through the target links. Clearly, the new paths must have enough bandwidth resources to handle the new traffic. In order to better exploit the available capacity, differently from other energy aware routing strategies that work at the level of traffic flows, in [113] the alternative path selection is performed at per-packet level. This allows a more accurate traffic splitting strategy to be defined which makes more efficient use of the available bandwidth, thus increasing the number of switched off links.

The authors in [114] propose an architecture which combines SDN with SR-based TE. An open source implementation of SR-MPLS is provided together with the realization of an SDN control plane which deals with the calculation of the optimal SR paths in the network. The authors begin implementing a basic TE heuristic, which solves in approximate way the flow assignment problem. The latter allows the overall network congestion to be minimized. This first procedure is also used as admission control for the next phase whereby the admitted paths are mapped onto SR paths using an heuristic of assignment (which has been described extensively in [15]—Section IV-E). Performance evaluation analyzes the distribution of path lengths comparing TE paths with the shortest paths and the distribution of the segment list lengths, thus showing that most of the paths can be implemented using 1 or 2 SIDs. All developed code is open source and available at [181]

A theoretical analysis of the computational complexity of the Traffic Engineering problems in Segment Routing enabled networks is provided in [115]. Two different TE problems are considered: i) the throughput maximization, and the ii) maximum link load minimization. As first the General Segment Routing paradigm is considered. In such a scenario, segments are not constrained to follow shortest paths, but can represent any (possible) complex path. The resolution of both the aforementioned TE problems results to be NP-hard. This finding provides a theoretical foundation to the reason why, in Segment Routing, shortest paths are considered for each segment. Following this, the complexity of the TE problems is studied for the case of Segment Routing with the shortest path.

An interesting outcome of this analysis is that, when the number of segments to be used for each segment list is fixed, the problem of minimizing the maximum link utilization can be solved in weakly polynomial time. Despite this, when the length of the segment lists is only upper bounded (not fixed), then the investigated TE problems fall again in the NP-hard class.

In [116], the authors deal with SR-based TE designing solutions for the optimal allocation of traffic demands using an ECMP-aware approach. The authors propose two optimal solutions for online and offline optimization using a 2-segment allocation, i.e., limiting the length of Segment Lists to two SIDs. The latter consists of the computation for each flow the optimal segment list of two segments with the objective of minimizing overall network congestion. The key idea of this work is to minimize the worst-case link utilization by considering ECMP forwarding in the offline case. While in the online case, the traffic split values are properly computed also to minimize rejections of requests. Performance evaluation shows that the n-segment routing problem ("Multicommodity Flow Problem"), i.e., with no constraints on the Segment Lists length, is just slightly better than 2-segment routing problem but the computation complexity is higher due to more degrees of freedom.

An extension of the models in [116] is proposed in [117]. In particular, the authors make use of the 3-segments forwarding demonstrating that the one defined in [116], using two segments, is not sufficient to determine the optimal paths and leads to wasting bandwidth.

DEFO (Declarative and Expressive Forwarding Optimizer) is a two layer architecture, described in [118], which is realized on top of a physical communication network, aiming at providing a flexible and highly programmable network infrastructure. At the bottom of the architecture there is a connectivity layer, which is in charge of providing default paths between the network routers. In DEFO, the connectivity layer is represented by an Interior Gateway Protocol (IGP). By means of an optimization layer, the routing paths of a subset of traffic demands is deviated by the default behaviour, provided by the connectivity layer, and is steered through a set of optimized paths. DEFO exploits the flexibility of Segment Routing to implement the optimization layer and configure optimized paths on top of underlying routing paths. The Service Provider can program the network, thus leveraging a high level interface that allows specific network goals to be defined through the use of Domain Specific Languages (DSL). DEFO makes possible for a network operator to define multiple cost functions to be optimized. In the basic case, referred to as "Classic Traffic Engineering," maximum link utilization is minimized; in the "Tactical Traffic Engineering" case, the objective function is a combination of the maximum link utilization and the number of modified paths. Once the goal has been specified, DEFO starts the computation of optimized paths, by running an algorithm that exploits the concepts of Middle-point Routing (MR) and Constraint Programming (CP).

The problem of reacting quickly to sudden traffic changes is faced in [119]. In fact, these unexpected events, which

occur at a low time scale, can create temporary congestion on links, thus degrading the network performance. Classic solutions to this issue are based on MILP models or Constraint Programming, see [116] and [118]. Unfortunately, these approaches suffer with high computation time, since they work on a time scale of seconds or minutes, providing TE routing strategies that on average allow a reduction of the network congestion, but that might incur in-link overloading due to sudden traffic spikes. For this reason, in [119] the authors present an algorithm which aims to mitigate link congestion under a hard time constraint. The proposed solution exploits SR to re-route a subset of flows quickly and flexibly in order to decrease the maximum link utilization. The time constraint is taken into account under two different perspectives: i) it is directly considered as a hard constraint during the algorithm execution, i.e., it is a termination condition, and ii) the selected routing strategy has to be as close as possible to the current one in order to minimize the number of reconfigurations needed to make it work. These two requirements are simultaneously satisfied by the proposition of a Local Search (LS) based heuristic, which takes as input an initial solution and iteratively goes from that solution to another one, by applying local changes called moves, until a stop criterion is met (e.g., the solution is good enough, or a time limit). Results show that the proposed algorithm overcomes MILP or Constraint Programming based heuristic, allowing for a significant reduction of network congestion with execution times lower than 1 second.

The Segment Routing Path variable is introduced in [120] with the aim of reducing the memory space and the computation time to formulate and solve TE problems in SR networks. In fact, while the memory space needed to instantiate classical TE problem formulations based on links, paths or nodes variables do not scale well with the size of the considered network and with the number of demands to be routed, SR path variables promise to increase efficiency in the problem resolution. SR Path variables are based on the concept of a Forwarding Graph (FG). An FG is a Direct Acyclic Graph (DAG) that originates from a node s and terminates on a sink node t. The path followed by a demand in the network is encoded as a sequence of FGs, and this sequence is stored into an SR path variable. Array-based sparse-sets are used to implement SR path variables. A Large Neighborhood Search approach is used to compute optimized paths for the demands. The idea is to iterative improve the best-so-far solution trying to reassign the value of a subset of SR path variables, related to demands that are critical (e.g., all the ones that are currently routed over the most loaded link).

The problem of migrating an IP network into a full SR-enabled one is investigated in [121]. The idea is that the process of upgrading the system of IP routers to enable SR capabilities is carried out incrementally in order to reduce the chance of introducing possible misconfigurations or causing the unavailability of the service. To do that, the Segment Routing Domain (SRD) is defined as the subset of SR capable nodes. Depending on whether the SRD is a connected set or not, two different models are proposed: Single-SRD or Multiple-SRD. Two main advantages of the S-SRD model

are that it limits the number of flow states to be maintained at the edges of the SRD, and the average length of the segment list is restrained. As a main drawback there is a potential decrease in the flexibility in the definition of network paths. On the contrary, M-SRD allows more complex paths to be defined at the cost of having a higher number of flow states and a higher average segment list length. The Segment Routing Domain Design problem is formulated as an ILP, where the main goal is to maximize Traffic Engineering opportunities, i.e., the identification of a subset of nodes, of a given size, to be upgraded with SR capabilities, so that the highest possible flexibility is achieved in balancing the links load in the network. The proposed formulation is able to capture both S-SRD and M-SRD models. The work shares several design principles with other works reported in this survey, for example it considers incremental deployments of SR and deals with path-aware encoding of the segments list in order to guarantee that the SR path will follow exactly the hop-by-hop path decided by TE heuristics. With respect to other works, the authors also propose a loose forwarding solution whereby the packets belonging to the same flow can cross the network using different paths.

ILP models and heuristics for TE applications in SR-based networks are proposed in [122]. Three ILP models are proposed and they are only used as a benchmark for the heuristics due to their high computational complexity. The first is able to leverage ECMP forwarding, the second one computes single routes and implements a hop-by-hop forwarding. Finally, the third one is able to leverage the full capability of Segment Routing. An heuristic has been implemented as some instances of the ILP models require too much time to be solved. The heuristic computes an unique route for each flow and tries to keep the total and the maximum network utilization as low as possible. Moreover, it is able to guarantee that the maximum value of the segment list depth is not exceeded.

A TE solution for path computation that, leveraging at most three labels, is able to optimize link resources and avoid congestion in the network is proposed in [123]. Firstly, the SDN controller addresses the problem of properly computing the weights of the IGP Link State protocol using Evolutionary algorithms. Then, the traffic distribution is computed through Distributed Exponentially-weighted Flow SpliTting (DEFT) or Penalizing Exponential Flow-spliTting (PEFT), which assign the flows to a next-hop with a probability that decreases exponentially with the extra length of the path (with the respect to the shortest path). SR is used to achieve detours and implement traffic splitting. Performance evaluation shows that the TE solution delivers a lower congestion with respect to OSPF/ECMP with optimized configurations and is able to use all available links.

The flexibility of SR in path selection, together with the higher throughput provided by the Multi Path TCP (MPTCP), are exploited in [124] in order to optimize the throughput for large flows and cope with the explosive growth of multimedia traffic in 5G networks. The proposed architecture uses a centralized control plane, with a central controller in charge of managing the Quality of Experience of each MPTCP

connection. In particular, the central controller finds multiple paths for each connection, checks the bandwidth requirements and installs specific flow rules at the ingress nodes of the considered 5G network. When a new MPTCP connection is established, the controller must find a path for each of the subflows belonging to this connection. For each subflow, it first checks the flow path and resource database, in order to check whether there is already a pre-computed path that has enough bandwidth to support the new subflow. If this check fails, the controller calculates a new path, encodes it into a segment list and installs a new flow entry in the ingress switch. The algorithm used to compute the new paths, named QoE-Centric Multipath Routing Algorithm (QoMRA), attempts to find multiple disjoint paths while considering QoE requirements.

The use of Multi Path TCP (MPTCP) in tandem with SR-MPLS to maximize the throughput of traffic flows in a Data Center network is investigated in [125] . MPTCP allows a single connection to be split over several paths, thereby increasing the total throughput. SDN-based MPTCP solutions are considered by the authors to achieve fine-grain control over TCP connections. However, this drastically increases the overall number of traffic flows that need to be stored in the devices, by causing scalability issue to arise. SR is used to reduce the number of flow rules needed to steer the single TCP connections over disjoint paths and to save precious space in the Ternary Content Addressable Memory (TCAM) of the devices. The architecture envisages a reactive approach for flow allocation: each time a new subflow is "generated" by MPTCP, a Packet-In is sent to the SDN controller which allocates a new disjoint path whenever is possible, and then installs the necessary flowrules to support this subflow in the edge devices. SR allows the state in the core device to be reduced. Unfortunately from the paper is not clear how the authors can avoid the explosion of the state at the edge of the network due to the matching conditions at the transport layer.

In [126], ELEANOR, a northbound application for the OpenDayLight (ODL) Software Defined Network (SDN) controller is presented. ELEANOR considers an MPLS-SR data plane, where a Maximum Stack Depth (MSD) constraint, i.e., an upper bound on the number of sids that can be stacked in a segment list, has to be considered. The main components of ELEANOR are: i) a path computation module, and ii) a label stack optimization module. When a new request arrives at the controller, it first finds a suitable path in order to meet specific QoS requirements (bandwidth, delay, etc.), then the appropriate SL is produced.

In [127], the authors propose two routing algorithms based on SR for realizing TE applications in SDN networks. These algorithms search for an appropriate selection of the link weights for optimizing path costs and balancing load across the links. This is obtained through the multiple objective particle swarm optimization (MOPSO) algorithm. Two objective functions are used to measure path cost and load balancing respectively. According to the authors, their algorithms not only reduce the cost of the paths, better balance the network load, and decrease the maximum link utilization rate, but also can improve the satisfaction ratio with respect of shortest path

first (SPF), shortest widest path (SWP), widest shortest path (WSP), minimum interference routing algorithm (MIRA), and the Lee algorithm.

The Bounded Stretch constraint to boost the resolution of the SR-TE problem is proposed in [128]. The Bounded Stretch is used to shrink the set of intermediate node candidates, which are selected during the building of the SL. This allows the space of the solutions to be reduced. The high level idea of the Bounded Stretch is that when an intermediate node is too far away from a source node i to a destination node j then this node should not be considered as a candidate. The selection is achieved comparing the length of the intermediate shortest paths with the length of the end-to-end paths scaled by a given constant. The authors demonstrate through experiments that the constraint helps in reducing the computation time at cost of having a slightly higher utilization over the links.

In [129], the node constrained TE problem is defined and analyzed, and SR is claimed to be an enabling technology for such routing strategies. This problem consists in the optimization of one of these two objectives: maximizing network throughput, or minimizing maximum congestion. Different routing strategies are considered. In the most general case, end to end paths are constrained to go through a set of middle-points, leaving the freedom to choose whatever path between two middle-points. This problem is formulated and proven to be NP-hard. Next, the feasibility region is limited by forcing the path between two middle-points to be the shortest one. The derived problem formulation turns out to be solved in weakly polynomial time. In any case, since the solutions of the previous problem can contain routing loops (the same link is crossed two times), the authors of [129] also consider a variant whereby the solution is constrained to select only acyclic end to end paths. This last variant of the node constrained TE problem is shown to be NP-hard. A further contribution of [129] is the proposition of the concept of *flow centrality* as a design parameter to select the most suitable middle-points. The flow centrality is expressed as the maximum percentage of traffic that can go through a node. This concept is further enhanced by defining the *group flow centrality*, which is a generalization of the flow centrality over a set of N middle-points.

The work in [130] proposes a traffic engineering solution (path computation and bandwidth allocation) for a hybrid IP/SR network able to maximize a utility function reflecting user satisfaction. User satisfaction is computed as a logarithmic function of the bandwidth assigned to flows. Routing paths are constrained to be the shortest ones in the IP domain, while SR routers can choose among a set of allowable paths in the SR domain. After defining the optimization problem, a two-step iterative algorithm is proposed: at each step of the iteration, link weights inside the SR domain are updated. Two main assumptions are made: i) each flow can be forwarded on a single path, and i) a packet cannot cross the SR domain more than one time.

Independently from the aim of the optimization and of the considered constraints, the main goal of a TE strategy is to find a routing configuration for a given set of input demands. Here we outline some of the differences between

the SR related research works classified in the category *Traffic Engineering*. Specifically, we point out five different criteria to compare them: i) the type of approach (optimization based, heuristic based, both), ii) the demand granularity level (Origin-Destination or Ingress-Egress), iii) the path computation strategy, iv) how the TE routing is translated into a set of SLs, and v) the traffic split policy (single SL or multiple SLs).

Most of the works on *Traffic Engineering* with SR propose a heuristic algorithm to determine a set of paths to satisfy a given objective. Only in [115], [116], [121], [122], [128], and [130] a problem formulation is also presented. Another interesting difference between the research works on *Traffic Engineering* is the model considered to describe the traffic demands. In fact, while the works in [114], [118], [119], [124], [125], [126], [128], and [129] consider an Origin-Destination (OD) model, where multiple demands can enter and leave the network from the same pair of Ingress-Egress (IE) nodes, all the others assume an IE model (there is a single demand between each pair of IE node that is representative of the aggregation of many OD flows). This affects both the flexibility of the solution, allowing for a thinner optimization, and the complexity of the algorithms, since they have to deal with a larger number of variables.

Different strategies are used to create the end to end paths. Some of them are based on the definition of parameters able to catch the current network status. The paths are then selected according to a Least Cost rule. As an example, in [109] the link "criticality" and the link congestion index are used, in [114] the concept of network crossing time is proposed, in [129] the feasible path space is reduced by imposing that only nodes with high centrality can be used as middle points. In [130] single path routing is considered, and the objective function of the optimization problem is the user satisfaction calculated as a logarithmic function of the bandwidth allocated to the user flows. Furthermore, also in [124] the path search strategy is based on the link criticality, while the concept of delay index is exploited in [125]. The remaining works use more sophisticated techniques, such as constraint programming ([118], [120]), ILP based search ([112], [113], [116], [117], [121], local search heuristic ([119]), and constrained Shortest Path First algorithms ([126]).

Another interesting difference of the path selection strategies adopted by papers falling into the *Traffic Engineering* category is related to the way they generate the SLs associated to the determined path. Specifically, two different approaches are generally used: i) the path is found and then encoded into a SL ([112], [121], [122], [124]-[126], or ii) the path is directly constructed as a SL ([116]-[120], [123], [128], [129]).

The final comparison that we propose is related to the possibility to split the traffic demands over multiple SLs. This option, which is allowed by properly configuring the SR policies, is explicitly used in the algorithms described in [116], [117], [121], [124], [125], [182]. Clearly, having the possibility to split the same flow over multiple SLs increases the flexibility of the routing strategy at the cost of increasing the information to be stored in the head end nodes, where SR policies are installed.

TABLE VIII
CLASSIFICATION OF THE REFERENCES RELATED TO FAILURE RECOVERY

Node failure	[132],[133],[134],[135],[136],[137]
Link failure	[131],[132],[133],[135],[136],[137] [138],[139]

C. Failure Recovery

Nine research works dealing with SR for Failure Recovery and Network Resiliency have been published in recent years. The proposed solution can be classified considering the type of failure they are able to recover from:, i.e.,link or node failures. Table VIII shows the classification of the covered papers. In the following paragraphs we provide a brief overview of the references classified as Failure Recovery related works.

Resilient SR forwarding is investigated in [131] . In particular, the authors focus on static fast failover solutions for Segment Routing, not requiring any interaction with the control plane. The algorithm proposed, referred to as Topology Independent Multi-Failure Alternate (TI-MFA), is an improvement of the Topology Independent Loop Free Alternate (TI-LFA), described in [45] and elaborated upon in Section III-A. TI-MFA has interesting performance guarantees and it is also resilient to multiple failures, while traditional SR fast failover based on TI-LFA can work only with a limited number of failures. Firstly, the authors demonstrate that TI-LFA loops indefinitely also for two link failures. Then, a robust but inefficient solutions is shortly presented which pre-compute routing rules considering destinations, incident failures and incoming ports of the packets. Even if this solution can provide better resiliency guarantees, it introduces some inefficiencies (in terms of path lengths) even if only one link failure occurs. Finally, the authors present their solution which basically proposes to store the already hit failures in the packet header and, each time a new failure is faced in the network, the segment list is re-computed using the pre-computed local table entries and the state stored in the packets.

Traffic duplication through disjoint paths is explored in [132]. In particular, the authors leverage SRv6 to realize a traffic duplication service which can guarantee an 1+1 protection scheme through the use of disjoint paths. The main difference from other protection mechanisms is that with 1+1 protection both channels are active and data is sent over both paths. The authors use mirroring behavior in the Linux kernel to realize the traffic duplication. The work builds upon the results presented in [183], leveraging the Linux implementation of SRv6. Then, the authors propose an algorithm that is able to compute disjoint paths with the least latency and that can be implemented with a number k of segments (they set an upper bound limit on their number and use only node segments).

The introduction of robustly disjoint paths is exploited in [133] and [132]. The authors built, extending routing theory and leveraging configuration synthesis, an automated compiler which is proactive, fast and self-healing (no external intervention are required): it computes pairs of disjoint paths for given sources and destination routers, which are robust in the way that they remain disjoint even upon a set of failures. This

is achieved without requiring any intervention thanks to SR technology. Indeed, SR allows sequences of segments to be written, which map to different disjoint paths even when there are topological changes. Finally in [133], leveraging the results of the solution proposed in [132], Aubry *et al.* added to the compiler the capability of limiting the number of segments (i.e., path encoding problem) and computing paths that do not degrade data-plane performance (finding low latency SR paths - addressing also TE aspects).

In [134], Hao and Lakshman propose a linear programming model to optimize the restoration in SR based networks. The key idea of the optimized restoration is to share the remaining bandwidth properly when several failures happen. This is addressed through an optimal configuration of the initial segments, knowing in advance the traffic matrix and the network topology. In particular, the authors develop an efficient primal-dual algorithm, which can handle single link failures and multiple logical link failures at the same time (including node failures). Moreover, with a simple randomized rounding scheme they can also take into account ECMP forwarding in the network.

A logically centralized implementation of the SR control plane (SDN based) is leveraged in [135], [136], and [137]. They describe a method to recover the network from link or node failures dynamically. The failover mechanism envisages a failover table for each interface of the node, and when a port goes down automatically the related secondary table is used to implement a loop-free backup path from the point of failure to the destination node. Firstly, the node pops all the labels in the segment list except the last label (which represents the final destination), and then the packet processing is passed to the correct failover table. The authors provide an implementation based on the OpenFlow (OF) protocol leveraging the OF Group tables feature for monitoring and backup actions (the use of OF Fast-Failover Group is explained in [184]). In [137] and in [136], the SR path encoding algorithm can lead to longer segment lists than the one generated by the algorithm proposed in [135]. In general, a low number of labels is necessary to implement most of the backup paths. The works in [137] and [136] implement a simple detour from the node detecting the failure towards the next-hop or the next-next-hop.

The same authors of the work in [108] propose an open source implementation of SRv6 TI-LFA in [138] using the extensible Berkeley Packet Filter framework (eBPF concepts in the Linux kernel are described in [185]). The fast rerouting solution is implemented as a custom BPF program compiled on the fly and attached to a route. In particular a program is loaded into the kernel for each link to be protected. The repair list associated to the route is computed by the control plane and then hard-coded in the eBPF program, which is subsequently compiled and installed in the kernel. The solution has been complemented with a robust failure detection architecture, which implements the Bidirectional Forwarding Detection (BFD) [186] echo mode using SRv6. In particular, the architecture envisages for each link a master node, periodically sending probes which can activate the fast rerouting mechanism described so far. The probes are sent with a special segment list which allows the redirecting of traffic to a special

BPF program (BPF slave) on the remote peer and the looping back of probes to the master. The BPF slave, running in the remote peer, handles the prob packets and can activate the SRv6 TI-LFA mechanism for its side of the link. The master uses SRv6 type-length-value to insert sequence numbers and timestamps, which also allow the remote peer to detect failures. The evaluation in the paper considers the number of false positives due to an overloaded CPU, the BPF implementation of the peer nodes reduces the false positives almost to zero, even when the failure detection is less than 10 ms. Instead, the master node is still largely affected by the overloading of the CPU since it uses an user space process for sending the probes.

The problem of defining a backup path scheme that is robust to the presence of multiple link failures is studied in [139]. Specifically, the main contributions are: i) the introduction of a polynomial-time fast rerouting algorithm which allows a backup path scheme for resilience to be defined under k simultaneous link failures, in particular the case of a hyper cube topology, and ii) the formalization, by means of an ILP formulation, of the problem of defining a backup path scheme that maximizes the number of allowed simultaneous link failures in general graphs.

Let us provide a comparison among the works classified in the *Failure Recovery* category. An implementation of the TI-LFA mechanism [45] in a Linux based SRv6 node is proposed in [138]. The TI-LFA mechanism is extended by adding the capability of handling multiple failures in [131]. Among the solutions based on SR to recover from network failures, there are only two works ([131], [139]) which are aimed at dealing with multiple failures. While the first is more focused on implementation aspects, the latter is more theoretical and focused on the definition of algorithms to find loop-free re-routing strategies. Not all other recovery mechanisms are explicitly declared as being able to deal with such a failure scenario. Another interesting difference between failure recovery solutions is the method they are based on. Specifically, most of them ([131], [135]-[139]) are based on a re-routing strategy, i.e., the packets are detoured over a pre-computed alternative path once a link is declared as failed. On the other hand, traffic duplication scheme is exploited in [132], [133]. Without going into the details, the main difference between works in [131], [139] and [135]-[138] is the number of simultaneous failures they allow recovery from.

D. Centrally Controlled Architectures

The definition of Centrally Controlled Architectures exploiting the SR architecture has been widely investigated in literature, resulting in sixteen different works. The solution proposed has been classified on the basis of three different aspects, as reported in Table IX. Considering the SR data plane, a high number of works makes use of the SR-MPLS data plane while only three works are based on SRv6. Moreover, some works do not explicitly consider a specific SR implementation, but simply exploit the SR possibility of inserting the flow state in the packet header. A further aspect used to classify the research papers is the protocol used for the

TABLE IX
CLASSIFICATION OF THE REFERENCES RELATED TO CENTRALLY
CONTROLLED ARCHITECTURES

SR data plane	SR concept	[146],[147],[151],[154]
	SR-MPLS	[140],[141],[142],[143] [144],[145],[150],[152]
	SRv6	[4],[148],[149],[153]
Southbound Interface	Openflow	[140],[144],[145],[147]
	PCEP	[141],[142],[143],[150]
	Other	[4],[150],[152]
Underlay data plane	IP/MPLS	[141],[142],[144] [145],[150]
	IP/SDN	[4],[140],[146],[147],[148] [149],[153]

southbound communication between the network devices and the central controller. The considered protocols are Openflow, PCEP or others. The last component that differentiates the proposed architectures is the underlay network. Specifically, the network devices can be IP/MPLS routers or Openflow switches. In the following we provide a brief overview of the references classified as Centrally Controlled Architectures related works.

An SDN based SR-MPLS architecture in a multi-layer packet-optical network is implemented in [140]. In particular, the authors demonstrate optical bypass upon traffic load variations without requiring GMPLS operations. The RYU SDN controller [187] has been extended to control the labels stack configuration at the edge nodes (Open vSwitch based). The SDN controller utilizes OF 1.3 to program the edge devices. Open vSwitch has been modified to increase the maximum MPLS stack depth from 3 to 15 labels. The OF protocol has been modified as well to push all the required labels as a single flow entry and with a single action. Commercial Reconfigurable Optical Add-Drop Multiplexer (ROADM) devices have been used to provide an optical bypass between nodes and provide alternative paths during path computation. In the performance evaluation, the authors asses through emulation the influence of the 15-label deep stack, evaluating the flows' setup time and packet forwarding in the devices. The latter is not influenced at all, while the setup time is almost triple.

Later in [141], SR has been implemented in a multi-layer network using a PCE architecture instead of SDN/OF. In this scenario nodes consist of commercially available IP/MPLS routers and the SR Controller is an extended version of a PCE stateful control plane. Extensions to the PCE protocol allow a centralized PCE to control the label stacking configuration. PCC (PC client - devices) uses the SR-PCE-CAPABILITY type-length-value for specifying the capability of handling SR-enabled Label Switched Paths and the capability to perform SR computation. The Explicit Routing Object (ERO) carried out in the Path Computation Reply message contains the list of computed SIDs and/or the Node or Adjacency Identifier depending on the SID type. On the device the agent collects the information derived from IGP protocol and configures the related shortest path entries and the SID labels. When a new PC replay message is received, the label stack is properly configured. Also this SR implementation using PCE is able to

perform dynamic packet rerouting (with optical bypass capabilities), by enforcing different segment-lists at source node, without any signalling protocol and with no packet loss.

In [142] Segment Routing is proposed as a solution to realize Network Service Chaining (NSC) in a metro-core network scenario, where service chain requests are represented by the so called micro flows, i.e., a huge number of low or medium bit-rate flows. In this scenario, classic solutions based on MPLS or pure SDN fail due to scalability issues. On the contrary, SR moves the flow state into the packet header, reducing the configuration costs (and time) to the installation of the encapsulation rule at the ingress point of the network (the one used to add the segment list to each packet of the considered flow). Based on this consideration, an SR Path Computation Element (SR-PCE) is defined in [142]. The SR-PCE is in charge of orchestrating connection setup/release/modification, and is made up of two main modules: i) the flow computation element, having the goal of find a path with available resources to serve a micro flow and ii) flow steering API that is responsible of installing the SR encapsulation rule in the ingress node. An experiment evaluation of the proposed architecture is proposed in [143].

An SR-based Software Defined Network (SDN) architecture is described in [144]. The proposed solution is able to perform load balancing among ECMP and non-ECMP routes in multi-layer networks including an IP/MPLS layer over an optical network layer. In particular, two solutions are described: i) Centralized-SR; ii) Preconfigured-SR. The former leverages a SDN controller to steer traffic over alternative paths upon network failures. Instead, the second solution uses OpenFlow load-balance groups to actively forward the traffic on several routes. With the second solution the data plane layer can autonomously react to a network impairment removing the failed output port, while the first solution always requires the intervention of the SDN controller, but results are more generic with regards of the second one. Both solutions push the SID of the destination node and leverage the available ECMP paths. The recovery property of the architecture has been validated simulating network failures. According to the authors, some packet losses have been recorded and the recovery time was around 170 ms.

The SPRING-OPEN project [145] is an ONOS [188] use case, which provides an open source SDN-based implementation of SR. Its architecture is based on a logically centralized control plane, built on top of ONOS, and it drastically eliminates the IP/MPLS Control Plane from the network. Part of this work converged later in the Trellis project [189], an SDN based leaf-spine fabric, built using bare-metal hardware, open-source software from the OCP [190] and ONOS projects, and OpenFlow-Data Plane Abstraction (OF-DPA) [191], an open-API from Broadcom [192] to program merchant-silicon ASICs. The leaf-spine fabric is based on SR-MPLS principles. However, it does not implement fully-fledged SR architecture, as it just uses global significant Node-SIDs. These MPLS labels are statically configured in the SDN control plane and are used to globally identify the ToR switches of the fabric, routing the traffic towards them using a single MPLS label.

In order to provide resiliency against link and node failures for Cloud Service Provides (CSPs), an overlay infrastructure realized by means of Segment Routing and Software Defined Networking control plane is proposed in [146]. The main idea of the proposed architecture is to substitute the dedicated physical infrastructure that interconnects Data Centers of a CSP, with an overlay network realized on top of an underlay infrastructure, represented by the interconnection of many Internet Service Provider (ISP) networks. The prerequisite is the availability of multi-homed connections for each Data Center of the CSP. Thus, the logical components of the proposed overlay infrastructure include: i) a central controller that monitors the underlay network status and determines new paths in case of failure, ii) the egress points that are responsible for routing the traffic flows leaving a Data Center toward the most appropriate ISP, and iii) the routing inflection points that are special nodes that manage the routing between two distant Data Centers, by using SR encapsulation.

The use of SR in a hybrid IP/SDN network as a technique to mitigate the problem of limited storage space in the flow tables of SDN switches, is presented in [147]. The main goal is to use SR to optimize the use of the flow tables and link capacity. In the considered scenario, every node supports both the OpenFlow operations and the normal IP forwarding operations. When a packet enters a node, it is first classified in order to decide through which pipeline it has to be steered, then it is processed accordingly. Operations attributed to the normal IP pipeline are decided using a routing protocol (e.g., OSPF). Alternatively, the path to be followed by traffic flows steered through the OpenFlow Switching Layer (OFSL) are decided by the central SDN controller. In order to limit the number of flow entries to install in order to configure a path, SR is exploited. In this way, a portion of the flow state information is moved from the flow table of the switches to the packet header. In order to insert the SR related information (i.e., the segment list) in the packet header, the use of unused fields (e.g., VLAN tag or optional fields) is proposed in [147].

Software Resolved Networks (SRNs) is a new SDN architecture recently proposed for IPv6 enterprise networks [148], [149]; further details about the implementation are provided in Section V. The network is managed by a logically centralized controller, which interacts with the end-hosts through an extended DNS protocol: applications are allowed to embed traffic and/or path requirements in their requests, and the controller is able to return the appropriate path to the applications satisfying their needs. SRv6 is used as data plane technology to steer traffic on a specific path according to network policies. Each component that can be reached through the network is always referenced through a DNS name. The default DNS resolver in the hosts is modified to interact with the controller of the architecture. SRN also provides a mechanism for the dynamic registration of the end-points. In this way, the DNS database can be properly updated and the name resolution can be performed by the clients. The connections are always unidirectional, thus it is necessary to establish two paths in order to enable the communication between the endpoints. A binding segment is used to implement a path id, and it is automatically translated in a SRv6 policy in the access node. Path segmentation is performed using the algorithm illustrated in [102], which allows a given policy to be matched and the minimal list of segments to be guaranteed. In order to optimize the interaction with the controller, upon a request the controller computes the two paths to support the communication and then instruct the access device of a node to add also the reverse binding segment in the SRH. In this way, the reply can be simply echoed back. Software Resolved Network has been implemented on Linux end-hosts, routers and controllers.

In [4] a novel SDN architecture is proposed for SRv6 technology; Section V provides further information about the implementation and where it is possible to download the code. The data plane is constituted by Linux based SRv6 nodes built from open source components which expose an open API towards the SDN controller. As a result, the nodes become hybrid as they envisage the coexistence of a legacy IP control plane with an SDN control plane. The authors present the design and implementation of the Southbound API between the SDN controller and the SRv6 devices, which is used to instantiate SRv6 policies in the network. Specifically, they propose a data-model and provide four different implementations of the API, respectively based on gRPC, REST, NETCONF and remote Command Line Interface (CLI). Topology discovery is also addressed by actively extracting the topology database from the IP routing daemons running in the network nodes.

A hierarchical multi-domain control plane for SDN networks based on SR has been demonstrated in [150]. The control plane is composed by an orchestrator application, which runs on top of multiple SDN controllers and leverages their NB APIs to create multi-domain SR based services. BGP-LS and PCEP are used as southbound in the SDN controllers. They provide network topology respectively and the creation of MPLS SR tunnels. IS-IS is used inside the domain to exchange reachability information and SIDs between nodes. SDN controllers do not exchange any reachability information nor SIDs. The orchestrator interacts with the SDN controllers and builds a global network view that will be used to perform the path computation and to instantiate SR services. A practical demonstration has been realized using software routers.

In [151] two solutions for multi-domain SR are proposed: end-to-end Segment Routing and per-domain Segment Routing. Both methods leverage a non-standard east/west interface between peer controllers, thus relying on a flat control-plane architecture and not using signaling sessions in the data plane. In the first approach, the segment list already contains the end-to-end path crossing several domains. The originator domain sends a request to the destination domain. The destination controller computes the segment list to reach the destination from its ingress router and sends it back to the previous domain. Each intermediate domain applies the same procedure stitching the segment list computed by the downstream until the reply is received back by the originator. Conversely, in the second approach the end-to-end path is obtained stitching several SR paths: in each domain the segment list contains a virtual label as the bottom of the stack,

which triggers a modification of the segment list in the ingress border node of the next domain. In this case, there is no global view of the network, and controllers do not know the domain sequence to reach the destination. Scalability of the proposed schemes is evaluated in terms of segment list depth. Results show that per-domain SR is able to encode 60% of the paths using at most 3 labels while end-to-end SR just the 12%. In general, the average SL depth is 5.34 and 3.36 respectively for end-to-end SR and per-domain SR.

In [152] an advance of Carrier Ethernet architecture is proposed and an approach mixing SR and SDN technologies is envisaged. It results in a trade-off between fully-distributed control planes and centralized approaches: an inventory database is maintained in the SDN controller with the configuration for each device. The SDN controller provides the IP configuration and SR configuration such as the loopback address, node label, label range, gateway label information via a southbound API such as NETCONF/Yang. For any communication inside the domain, the network will use the IGP based forwarding by design without the need of the SDN controller and will impose on the traffic a single MPLS label (loopback SID). Multiple labels can be used to realize TE applications. Instead, in a multi-domain scenario several SDN controllers are deployed and exchange reachability information in order to properly program the edge nodes. In this way, an inter-domain path can be established simply with a label stack that includes local and remote border router labels, plus the end node label. Similarly to other works, further analysis is not possible as the code of the control plane is not open source. As for the data plane, the solution relies on Carrier Ethernet hardware.

Busoni [153] is an orchestration framework for Segment Routing based networks, which automates and simplifies many aspects of the network management. From an architectural standpoint, Busoni sits on top of a SDN controller and benefits from the information exported by the controller to feed its data-store. The latter is a graph database, which is used to persist data. In particular, Busoni leverages it to keep track of the SIDs advertised in the network, the installed policies, and to respond to any dynamic event. The framework provides users with programming tools to compose and manage SR policies . It operates efficiently even under multi-tenancy environments. Finally, Busoni automatically updates the nodes and the edges of the graph database whenever there is an update in the network and reflects these changes on the installed policies. This allows their resilience to dynamic events to be maintained.

A scalable centralized controlled architecture for the management of SFC Routing and Cloud Bandwidth resource Allocation (SRCBA) based on SR is described in [154]. The proposed solution is thought to be applied in a multi domain scenario, where a transport network interconnects a set of private cloud infrastructures, possibly owned by different providers. In particular, instead of using classic approaches to solve the SRCBA problem, which require a detailed knowledge of either the transport network or the cloud infrastructure, the proposed architecture exploits the BSID concept to abstract the services provided by a single cloud infrastructure to the

external network. The resulting Orchestrator is then divided into two logical components: i) a centralized Network Service Orchestrator (NSO), which is in charge of collecting SFC requests and managing bandwidth in the transport network, as well as deciding to which datacenter to assign the processing of the incoming requests, ii) and a set of local Resource Orchestrators that are in charge of managing the network and cloud resources in the context of a single infrastructure. In this way, the centralized NSO can rely on summarized information to take decisions while solving the SRCBA problem. This allows for a great reduction of computation time, while assuring comparable performance in terms of efficiency in the use of the resources.

In the following paragraphs we provide a comparison among the works classified in the *Centrally Controlled Architecture* category. An interesting aspect to compare the research works falling into this category is related to the way they use SR. For instance, some solutions exploit SR to overcome some limitations existing when other technologies are adopted. Other centrally controlled architectures make use of SR to realize specific functions, which are more complex if realized by means of other paradigms. Finally, some works falling into this category have the goal of proposing an implementation, and provide a demonstration of the feasibility and the performance achieved.

Several works [142], [147], [153], [154] exploit SR to overcome limitations of other existing approaches. Specifically, in [142] SR allows for the reduction in the complexity of configuring and updating the path for an incoming SFC request, thanks to the adopted source routing paradigm. In [147], they exploit the same principle to show that, by adopting SR, the stringent TCAM size constraint can be overcome. Finally, in [153] an orchestration framework to simplify the policy management is proposed. Similarly, a scalable centralized controlled architecture for the management of the SFC Routing and Cloud interconnect bandwidth allocation is defined in [154].

Different functions at the network level are realized in centrally controlled architectures proposed in [144], [146], [151]. As an example, in [151] SR is used to realize an optical bypass, while it is exploited to overcome node and link failures in [144]. Finally, an overlay network to interconnect geographical distributed DCs is realized through SR in [146].

As previously stressed, the aim of some of the works falling into the *Centrally Controlled Architecture* category is to propose possible implementations of SR, and to provide a demonstration. This is the case in [4], [145], [150]. In [4], SRv6 nodes based on the Linux implementation are considered in the data plane, while the other works are focused SR-MPLS. Furthermore, a demonstration of end to end paths setting in a multi domain SR network is given in [150], while in [145] a single domain scenario is considered.

E. Path Encoding

The translation of a network path, resulting from a specific TE objective, into a sequence of SIDs, i.e., a Segment List, is a key operation for the deployment of SR in a real network.

TABLE X
CLASSIFICATION OF THE REFERENCES RELATED TO PATH ENCODING

Path to be encoded	*uniform IGP weights*	[155],[156],[157]
	arbitrary IGP weights	[158],[15],[159],[160],[161]
	ECMP not allowed	[15],[155],[157],[158]
	ECMP allowed	[156],[159],[160],[161]
Requires further configuration	*no*	[155],[156],[157] [158],[15],[159]
	yes	[160],[161]
Encoded path	*single SL*	[155],[156],[157],[158] [15],[160],[161]
	multiple SLs	[159]

This operation is usually referred to as Path Encoding. Eight different papers have been defined for the definition of a proper a solution to the path encoding problem. The different algorithms differ according to the nature of the network path, i.e., the path to be encoded. Specifically, the input path might be computed on top of a network with uniform link weight or arbitrary ones. Moreover, it may or may not include ECMP. A further aspect for the path encoding algorithms is the possibility of requiring additional device configurations, as the insertion of a new policy. Finally, path encoding algorithms can differ depending on the possibility of encoding the input path with one or more SLs. Table X shows the paper classification according to the aforementioned aspects. In the following we provide a brief overview of the references classified as Path Encoding related works.

In [155], SDN and PCE based implementations of the SR controller share a common path engine, which performs the hop-by-hop path computation and SR path assignment. As regards the path computation engine, the controller selects the least congested path on a set of candidate paths. Then, the proposed SR path assignment algorithm provides the shortest Segment list considering a unique path towards the destination and avoiding load balancing through ECMP. The algorithm uses two pointers, i and j, to navigate the target path from source to destination. Firstly, j is incremented until the considered sub-path is no more the unique shortest path. At this point, the SID of the node $j - 1$ is inserted into the Segments list, the two pointers are both set to $j - 1$, and then the procedure restarts. This cycle is repeated until the node j is equal to the destination. The algorithms provide the segment list of minimum depth, but the solution only considers global Node-SID, and therefore it cannot be applied to topologies with arbitrary IGP link costs.

The authors of [156] propose a Segment list encoding algorithm to express a given path, which minimizes (enforcing a given threshold) the Segment list depth in SR-based networks. It considers ECMP forwarding by default, but can also introduce constraints to support a deterministic hop-by-hop path. With respect to other efforts (such as [15]), the solution is not able to support arbitrary hop-by-hop paths when arbitrary IGP link costs are used. The core of the algorithm consists of the creation of a graph, whose arcs model SR related instructions (node and adjacency SIDs). Specifically, an arc connecting

two nodes represents the shortest path in the original network between the same pair of nodes. The so-called auxiliary graph is built firstly using the physical links of the paths, which are computed by an external TE heuristic. Subsequently, virtual links are added representing the ECMP paths between two nodes. Each virtual link is annotated with the metrics and the number of the ECMPs between the two nodes. Using this auxiliary graph, a new path computation is performed adding a new constraint related to the number of hops: each path having a hop number greater than the maximum Segment list depth is rejected. The candidate paths are then sorted firstly according to their original metrics, secondly according to their length, and then according to the number of ECMPs. Finally, the paths are translated into SIDs using this approach: physical links are changed with their respective Adjacency SID, virtual links are mapped with the Node SID of the destination node. If the Adjacency SID are not local the algorithm first inserts the Node SID of the source and then the Adjacency SID.

In [157], two algorithms for the computation of the segment lists are proposed; the algorithms are optimal in terms of stack depth when a unique hop-by-hop path has been computed. The key idea is to consider at each iteration bigger sub-paths and verify whether a unique shortest path exists. It it exists, substitute the sub-path with the tail node SID before moving to the remaining part of the path. The main difference between the algorithms is how to vary the breadth of the sub-paths until considering the original hop-by-hop path. The first algorithm navigates the target path starting from the source node toward the destination, while the second one leverages the opposite direction. The analysis of the overhead in the packet headers shows that reverse algorithm introduces less overhead compared to the direct algorithm, as the computed segment list typically includes nodes that are near the source. Other works (for example [15], [114]) share the same objective of reducing the overhead of the packet headers, but with respect to the solutions described above, they start from the original hop-by-hop path and then evaluate the sub-paths breadth reduction at each iteration.

In [158], two algorithms for an efficient paths encoding are proposed. The algorithms, referred to as SR-LEAs, take as input an explicit shortest path and then compute the relative segment list having as constraint a given maximum segment list depth. They are composed of two main steps: i) computation of successive shortest paths; ii) label replacing. Specifically, they compute the subpaths of the original shortest path and take into account the limitation of the hardware to reduce the number of subpaths. In the second step, the subpaths composed of three or more nodes are replaced by the Node-SID of their tailnode. SR-LEA replaces two node subpaths using the Adj-SID. The variant SR-LEA-A is very similar to the above algorithm but takes advantage of the global Adj-SID to further reduce the depth of the label stack. Of course, it requires the advertisement of these SIDs in order to work properly. Simulation results show that the algorithms are able to compute segment lists with an average length lower than 3. SR-LEA-A delivers best results and can improve the segment list computation compared to SR-LEA, but the gain is around 5% in terms of average length.

The authors of [15] propose an optimal SR path assignment algorithm and prove that it is optimal in terms of the number of used segments. The algorithm takes as an input a hop-by-hop path resulting from a TE algorithm and define an SR path for it composed by the minimum number of segments. The SR assignment algorithm evaluates at each iteration if a shortest path between the current source and the current destination exists, considers different subpaths and updating the current source or the current destination. Each time a new segment is found, the current source is updated with the previous destination, and the current destination is replaced by the destination of the original hop-by-hop path. In the worst case, the link between the current source and its next hop is evaluated, and if the link is not part of the shortest path, an adjacency segment is pushed in the segment list.

The authors of [159] propose a two-step method for translating a given Traffic Engineering (TE) path into an SL. In the first step, an auxiliary graph is created. The aim of the auxiliary graph is to represent all the Interior Gateway Protocol (IGP) paths available, i.e., forwarding paths, for the specific TE path to be encoded. In the second step, a MILP problem over the auxiliary graph is defined: the MILP problem allows for the minimization of the overall Segment List length given the target TE path. One of the main features of the proposed solution is multi-path support, i.e., the ability to split a source-destination flow among a weighted set of segment lists.

The problem of optimizing the performance of an SR network under the maximum Segment List Depth (max-SLD) constraint is studied in [160]. In fact, especially when SR is realized on top of the MPLS data plane, the constraint on the max-SLD is particularly limiting. A possible solution is to create new LSPs, thus increasing the availability of alternative paths in the underlay network, and consequently enabling shorter segment lists to be written. Despite the fact that the creation of new LSPs reduces the length of the segment lists, its main drawback is the increase in the number of required forwarding rules to be installed in the routers' forwarding tables. In order to mitigate this negative effect, the concept of panel based forwarding is defined in [160]: a panel refers to a set of node-disjoint LSPs that can be represented by the same label. Furthermore, an ILP formulation is proposed to solve the path encoding problem, minimizing both the number of new defined LSPs and the installed rules, while respecting the max-SLD constraint.

To overcome the Maximum Stack Depth (MSD) constraint in SR-MPLS, a new type of SID, named Targeted SID (TSID), is defined in [161]. A TSID is a local segment that is associated with a sequence of SIDs. The instruction related to a TSID consists of replacing it in the SL with the associated sequence of SIDs. By using a TSID, it is possible to reduce the length of an SL at the cost of introducing a new flow state in the node that implements the instruction related to the TSID. For this reason, the authors of [161] propose two different optimization problems that allow a trade-off between the benefits and the costs of the TSID tool. The first optimization problem takes as input the set of paths with an SL overcoming the MSD bound and aims to minimize the number of defined

TABLE XI
CLASSIFICATION OF THE REFERENCES RELATED TO NETWORK PROGRAMMING

Service Function Chaining	[21],[162],[163],[164],
Operational Function	[165],[166],[167],[100]

TSIDs. The focus is then the reduction of the number of extra flow states to be maintained by the network nodes. A second optimization problem is presented with the goal of minimizing the PCEP sessions that have to be maintained between the central controller (responsible for the TSID installation) and the nodes where the TSIDs have to be installed. In this case, the main idea is to install as many TSIDs as possible in the same node.

A possible key to comparing the existing SL encoding techniques is to focus on the method they use to obtain the final list of SIDs. In particular, two different methods are exploited. The first one is based on the Bellman-Ford principle, which states that each sub path contained into a shortest path is in itself a shortest path. Based on this consideration, these algorithms explore the path to be encoded, starting from the source node up to an intermediate one. Until the path between the source and the intermediate node is found to be a shortest path, they move on by considering the next node in the path. When this condition does not hold, the first SID is found. The process goes on until the full path is explored. SL encoding tools presented in [15], [155], [157], [161] use this approach. Furthermore, it is interesting to emphasize that all the works using this approach do not allow the use of ECMP in the underlay.

The second type of approach consists of the creation of an auxiliary graph to represent the underlay paths. Specifically, an arc of the auxiliary graph is an entire path in the underlay. This type of approach is used in [156], [159]. The main difference between the solutions presented in these two works is that, in [156] the end to end path is encoded by applying the Dijkstra algorithm over the auxiliary graph, while in [159] the problem is modeled as a Multi Commodity Flow over the auxiliary graph. As a consequence, the use of multiple SLs to steer a single flow is allowed also in [159].

F. Network Programming

Despite the fact that Network Programming is the most attractive and innovative feature of SR, we have found only eight papers related to this topic. As shown in Table XI, these works can be classified in Service Function Chaining related ones, or operational function. The latter are aimed at implementing operational functions, such as a firewall, a load balancer and a zero-loss VM migration tool, by exploiting the network programming feature of SR. In the following paragraphs we provide a brief overview of the references classified as Network Programming related works.

In [162] the Linux SRv6 implementation described in [183] is enhanced introducing the support for Service Function Chaining. In more detail, the Linux kernel provides an API to map a service segment, i.e., the identifier of a network service

running on a Virtual Machine. The experimental evaluation shows that the impact of SR operations and service segment processing on the packet forwarding capabilities is limited, i.e., with a reduction lower than 10%.

The architecture of a network domain supporting Service Function Chaining (SFC) through SRv6 is investigated in [163]. The authors mainly focus on the implementation of a VNF node able to host multiple VNF instances. The main components of the VNF nodes are the SR/VNF connector, in charge of logically connecting the SR routing with local VNFs, and the VNFs, supporting specific network functions. The VNFs can be SR-aware or SR-unaware: i) SR-aware VNFs can process the information contained in the SR header (SRH) of incoming packets; ii) SR-unaware VNFs are not able to handle the SRH. Thus, in order to correctly apply the VNF to the original packet, the SR/VNF connector must pre-process the packet by removing the SR encapsulation, and then re-apply it when the packet is returned by the VNF. The authors propose a Linux-based implementation of a VNF node supporting both SR-aware and SR-unaware VNFs. In more detail, using the netfilter framework, a new kernel module called srext (Segment Routing EXTensions) is implemented to act as a SR/VNF connector and to support SR-unaware VNFs. A virtualized testbed based on Virtualbox and Vagrant is realized to evaluate the processing overhead introduced by the proposed implementation with respect to a classic IPv6 forwarding solution.

SRV6Pipes [164] is an extension of the IPv6 implementation of Segment Routing in the Linux kernel, which enables chaining and operation of in-network functions operating on streams. SRv6 is used to enforce an end-to-end path between the client and the server passing through the equipment hosting the networking functions. The rationale behind SRv6Pipes lies in the fact that some network functions need to include a TCP implementation to work on the streams. SRv6Pipes leverages "the TCP stack that is already present in the Linux kernel" and implements a transparent TCP proxy to offload the TCP functionalities to it and terminate the TCP connections where a network function is deployed. In this way, it is possible to expose to the network functions the bytestreams they need to process. Special addressing is used to specify network functions and their parameters. In particular each proxy exposes an 80 prefix. The first 80 bits are used to traverse the proxy, the following 16 bits are used to identify the network function and the remaining ones are used to specify the parameters of the function.

The concept of SR Aware VNF, as an application that is able to process the SR information in the packet, is defined in [165]. In the same work the implementation of an SR Aware (SERA) Firewall application is also proposed. The SERA Firewall is able to work both as a legacy firewall (basic mode), or define filtering rules that also include condition on the SR fields (advanced mode). In the basic mode, it can apply the normal firewall processing to the original packets even if they have an SR based SFC encapsulation. It means that the filtering rules are applied to the original values of the packet header (the SR related fields are transparent). In the advanced mode, SERA

offers new matching capabilities and new SR specific actions that allow fields in the SR Header to be modified.

Segment Routing Load Balancing (SRLB) [166] is an Application-aware load-balancer that avoids cost due to monitoring tasks. It is thought to work in the context of a Data Center network, where several instances of the same application are instantiated in different host machines. Each host machine is equipped with a VPP based virtual router which dispatches packets between physical NICs and application-bound virtual interfaces. The Load Balancer (LB) is located at the edge of the Data Center network. A request for an application is represented by the first packet sent from the client to the application server (generally a syn message). When a new request arrives at the LB, the Service Hunting function, which consists of finding a server that can serve the current request, is executed. LB exploits SR to query a subset of servers that host the requested application. Specifically, LB encodes the set of randomly selected potential servers in the segment list before encapsulating the first packet of the request. When the first server receives the packet, it can decide whether to deliver it to the application, and consequently assign to it the processing of this request, or it can forward the packet to the next server in the segment list. The decision about whether to accept/refuse a request is taken according to a connection acceptance policy which takes into account the internal state of the application (CPU usage, memory usage, etc.).

A new architecture for the Content Delivery Networks (CDNs) building upon the results of [166] is described in [167]. In this new paradigm, CDN decisions (cache vs origin servers) are offloaded to the data plane. This architecture is based on two fundamental pieces: i) chunk-level content addressing and ii) in-network server selection. The first is realized by assigning a unique and globally routable IPv6 address to each chunk. Instead, the in-network server selection leverages these identifiers exposed as IP addresses to make in-band forwarding decisions, which are later bound to a SRv6 steering policy. In particular, upon arrival of a request at a CDN proxy, the IPv6 identifier is used by the prediction engine to perform cache admission by estimating the popularity of requests with a Least-Recently-Used (LRU) filter. If this is not available (cache miss), 6LB [166] is used to forward requests directly to the origin servers instead of proxying them at the cache. According to the authors, this mechanism allows the load on the edge cache to be reduced and negative effects on the Quality of Experience to be avoided.

An SR based live migration technique achieving zero packet loss is proposed in [100]. The starting point is an SRv6 enabled Data center network. Two new SR functions are defined and used in the process of VM live migration. The first one is *forward to local if present*, which is a conditional version of the END behavior described in Section II-C. Specifically, in case the last SID in the SL is locally available, then the packet is directly processed, thus ignoring possible intermediate SIDs. The second one is *buffer and forward to local*. This forces the node to inspect the last SID, and if it is not locally available, the packet is buffered until the SID becomes available. Assuming that a VM is migrated from host A to host B, all

requests received by the Data center gateway are then tunneled during the migration process using an SL whereby: i) the first SID points at the *forward to local if present* function implemented at A, ii) the second SID points at the *buffer and forward to local* function implemented at B, and iii) the last SID is referred to the VM. In this way, until the VM is available at A, the packets are directly delivered to the VM thanks to the *forward to local if present* function. Then, during the downtime, the packets directed to the VM are buffered at B thanks to the *buffer and forward to local*. Finally, when the VM becomes available at B, the buffered packets, as well as new arrivals, are directly delivered to the VM at B. In this way it is possible to implement VM live migration having no packet loss.

SRNK [21] is an SR-proxy for legacy VNFs which are unaware of SRv6 technology and expect to process traditional IP packets. SRNK extends the implementation of SRv6 in the Linux kernel [193] adding the support for the *End.AS* and *End.AD* behaviors. The performance of the proposed solution has been evaluated by the authors, who identified a poor scalability with respect to the number of VNFs to be supported within an NFV node. With an enhancement of the Linux Policy Routing framework, they provided a second design SRNKv2, which does not depend on the number of supported VNFs in a node. They compared the performance with a reference scenario not performing the encapsulation and decapsulation operation and demonstrated that the overhead of SRNKv2 is very small, on the order of 3.5%.

The Service Function chaining topic is investigated in [21], [162], [163], [164]. At the same time, there are some important differences that have to be highlighted. An API in the SRv6 implementation of the Linux kernel to map the service segments is defined in [162]; it mainly deals with SR-aware VNFs. Instead, a SR-proxy for SR-unaware VNFs is proposed in [21]. The work in [163] implements a solution able to deal with both SR-aware and SR-unaware VNFs, but with respect to the previous works it extends the netfiler framework, while the previous solutions are extensions of the SRv6 implementation in the Linux kernel. Finally, a further extension of the SRv6 implementation in the Linux kernel is described in [164], but it enables the chaining of in-network functions operating on streams, thus working at a higher level (Transport layer) with respect to the previous works.

G. Performance Evaluation

In this subsection, we report four papers dealing with the performance evaluations of SRv6 implementations.

SRPerf [168], [171] is a performance evaluation framework for software and hardware implementations of SRv6. SRPerf is able to perform different benchmarking tests such as throughput and latency. At the time of writing, the framework supports Linux kernel and VPP implementations of SRv6. The architecture of SRPerf can be easily extended to support new benchmarking methodologies as well as different SRv6 implementations. The framework supports two different metrics to characterize the throughput of a SRv6 enabled node: No-Drop Rate (NDR) and Partial Drop Rate (PDR). PDR is defined

as the highest throughput achieved without dropping packets more than a predefined threshold. NDR, which corresponds to the Throughput defined by RFC 1242 [194], can be described as PDR with a threshold of 0%. The framework orchestrates all aspects of an experiment starting from the setup of the testbed to the enforcement of SRv6 configurations, thus relieving the experimenter from a significant configuration effort. SRPerf has been used to evaluate the performance of SRv6 implementations in the Linux kernel and in VPP. Moreover, in [171], the authors propose the evaluation of an enhanced Linux kernel, which has been obtained by adding the implementation of missing behaviors and fixing the implementation of existing ones.

A solution whereby low-level network functions, such as SRv6 encapsulation, are offloaded to Intel FPGA programmable cards is presented in [169]. In particular, the authors partially offload the SRv6 processing from a VPP software router to the NICs of the servers, thus increasing data-path performance and at the same time saving resources. These precious CPU cycles are made available for VNFs or for other workloads in execution on the servers. Tests results of the *End.AD* use case show in the worst scenario a CPU saving of 67.5%. Moreover, the maximum throughput achievable by a pure VPP solution with 12 cores is obtained by the accelerated solution using only 6 cores.

The use of SRv6 as alternative user plane protocol to GTP-U [99] is studied in [170]. Firstly, the authors propose an implementation of the GTP-U encap/decap functions and of the SRv6 stateless translation behaviors defined in [24]. These behaviors guarantee the coexistence of the two protocols, which is crucial for a gradual roll-out. The authors used programmable data center switches to implement these data plane functionalities. Since it is hard to get telemetry from a commercial traffic generator when a translation takes place, the authors injected timestamp with a resolution of nanoseconds to measure the latency of SRv6 behaviors. Finally, they measured throughput and packet loss under light and heavy traffic conditions in a local environment. Results show no huge performance drop due to the SRv6 translation. Moreover, the latency of the SRv6 behaviors is similar to the GTP-U encap/decap functions.

H. Miscellaneous

In this section, we have included all works not belonging to previous categories. In general, these works are very different among them and have only Segment Routing in common.

A tutorial on Segment Routing and a survey on research activity covering more than 50 scientific papers is provided in [14]. The tutorial part focuses on the SR-MPLS data plane part and does not consider the recent SRv6 data plane. SR standardization efforts, implementation activities and deployments are not specifically analysed in [14].

The Control Exchange Point (CXP) defined in [172], even if not explicitly based on SR, proposes the concept of *pathlets* [173], which closely resembles the idea of the list of instructions present in the SR architecture. The main goal of Control Exchange Point (CXP) is to provide services with

QoS constraints across domains. This is achieved stitching the *pathlets* which are partial paths advertised by domains. An ISP abstracts its network as a set of *pathlets* connecting the network edges, and then advertises these on the northbound. More specifically, this abstraction is realized with tunnels instantiated with OF, MPLS, optical paths and so on. The *pathlet* abstraction is bundled with properties that the ISP provides such as latency, costs, available bandwidth and so on. A CXP is an external entity acting as brokering layer and providing inter-domain routing coordination based on SDN APIs. The general idea seems to be in line with SR architecture presented so far, and it can be implemented using SR data-plane technologies.

An alternative implementation of SR architecture through Omnipresent Ethernet (OE), which is a modification of Carrier Ethernet architecture, is proposed in [174], [175], and [176]. It is based on source-routed, binary-routed labels embedded in an Ethernet frame. Details on the implemented SR header are provided in [176]. The authors of the aforementioned works address scalability issues of SR in the context of multi-domain scenarios from two different points of view: packet header size and the number of table entries required at the edge nodes. They realize also a testbed to validate the implementation of the proposed schemes. However, further analysis is not possible since the solution builds upon Carrier Ethernet hardware.

The TE problem for SR is analyzed from a different perspective in [177]: the authors evaluate the influence of the metrics used to define the cost of the links. Key findings of the study indicate that the routing metric can influence shortest path based TE. Very simple and reasonable metrics such as inverse capacity work as well as a complex optimized metric. They allow this to be close to the optimum with respect to the most common traffic engineering objective of minimizing maximum utilization. Finally, if other objectives are introduced besides optimizing link utilization, the choice of the metric is more important and there are significant differences between the metrics. Although the work deals with Traffic Engineering, it does not provide a new heuristic nor a different way to address the problem, rather it is simply a study on the influence of the routing metrics. Therefore we have not classified it in the TE category.

In [178], the authors take advantage of Segment Routing to build an SR-based Multicast delivery mechanism to efficiently provide live video streaming services for 5G users. SR is used to alleviate the rule update overhead and to avoid the explosion of routing tables in the devices. The work builds upon an interesting problem, although it does not provide a lot of details about SR implementation. The work mainly focuses on the problem of building an efficient Multicast tree to further reduce the rules update due to the user movements (Handover-aware Multicast tree).

An Industrial IoT (Internet of Things) scenario with an extremely large number of objects to be connected, also taking into account their mobility, is considered in [179]. The proposed solution relies on Segment Routing for enabling scalability and flexibility in packet forwarding. This is in particular to bypass the overloaded links and to achieve load balance.

A novel distributed processing model for IoT, based on the extension of the SRv6 Network Programming Model, is proposed in [180]. The idea is that each IoT node offers an abstract machine that can be programmed using an Instruction Set Architecture. The program can be embedded in an SRv6 segment list. An SRv6 packet carries both the program and the execution state. It can travel across IoT nodes, reading and writing the I/O ports of the device and executing computations as dictated by the program in the packet itself.

V. SR Implementations and Deployments

In this section, we describe the implementation results related to SR. We will mostly focus on the SRv6 version, which is attracting a lot of interest and development efforts. The SR-MPLS version is already in a mature development phase, well supported by the main core router vendors (e.g., Cisco, Huawei, Juniper). SR-MPLS can be incrementally deployed in current IP-MPLS backbones, as it only requires software updates to networking devices. Operators can migrate to SR-MPLS to simplify the control plane operations and improve the scalability. As for the SRv6 data plane, there are two main Open Source data plane implementations for software routers: the Linux kernel implementation (described in Section V-A) and the implementation done by the FD.io VPP project (described in Section V-B). Section V-C presents other open source implementations, mostly related to research activities. Finally, in Section V-D we briefly analyze the hardware implementations of SRv6, the inter-operability efforts done by several vendors and the current deployments of SRv6 in large production networks.

A. Linux Kernel

The SRv6 capabilities were first added in Linux kernel 4.10 [193]. Kernel 4.10 includes the support for some *SRv6 Policy Headend* behaviors (formerly known as *transit* behaviors) (e.g., *H.Insert* and *H.Encaps*). The *SRv6 Policy Headend* behaviors are implemented as Linux Lightweight Tunnel (LWT). The implementation of the iproute2 [195] user space utility is extended to support adding a *localsid* associated with an *SRv6 Policy Headend* behavior [196]. SRv6 *localsids* with *SRv6 Policy Headend* behavior are added as IPv6 FIB entries into the kernel main routing table. Kernel 4.14 is another important milestone for the SRv6 support in Linux: a set of SRv6 *endpoint* behaviors have been implemented by adding a new type of LWT [197]. The supported SRv6 *endpoint* behaviors are *End.X*, *End.T*, *End.DX2*, *End.DX4*. *End.DX6*, *End.DT6*, *End.B6*, and *End.B6.Encaps*. Some new *SRv6 Policy Headend* behaviors have been added (e.g., *H.Encaps.L2*). The iproute2 implementation was extended as well [195], [198]. The SRv6 capabilities in Linux kernel were extended in kernel 4.16 [199] to include the netfilter framework [200]. A new iptables match extension, named `srh`, was added to the kernel to support matching of SRH fields. The `srh` match extension is a part of the SERA firewall [165] and supports matching all the fields of the SRH. The implementation of iptables user space utility [201] is extended with a new shared library

(`libip6t_srh`) that allows to define iptables rules with `srh` options.

Linux Kernel 4.18 [202] has added few more features both in the core SRv6 stack and in the netfilter framework. In the netfilter framework, the `srh` match is extended to provide the matching of SRH's *Previous SID*, *Next SID*, and *Last SID*. The iptables user space utility is updated as well to support the new matching options. Instead, a new feature is added in the Linux SRv6 stack to support custom SRv6 network functions implemented as small eBPF [185] programs. An extension of [3] is provided in [108], with the introduction of a new *End* behavior, i.e., the so called *End.BPF*. From an implementation point of view, a new hook for BPF is added to the SRv6 infrastructure that can be used by network operators to attach small programs written in *C* to SRv6 SIDs which have direct access to the Ethernet frames. Moreover, specific SRv6-BPF helpers have been provided in order to allow *End.BPF* functions to execute basic SRv6 actions (*End.X*, *End.T* and many others) or adding TLVs. This allows custom *SRv6 Policy Headend* behaviors to be implemented (mainly to extend SRv6 encapsulation policies implemented by the kernel). The tutorial about eBPF extensions to SRv6 is available at [203]. The source code of the sample applications described in [108] is freely available at [204]. Instead, the eBPF-based fast-reroute and failure detection schemes described in [138] is available at [205].

SR-MPLS has not received the same attention of the SRv6 implementation in the Linux kernel. All the features which are available are mostly related to the well-established MPLS forwarding. They have been made available from the version 4.1 of the kernel. In particular, kernel v4.1 has seen the introduction of the MPLS Label Switching Router (LSR) behavior. MPLS capabilities have been extended later in the kernel v4.3. LWT framework and MPLS tunnel were added allowing the implementation of the MPLS Label Edge Router (LER) behavior. Finally, MPLS multipath functionality has been added only in the version 4.5 of the kernel.

In general, the Linux kernel lacks of the support of the SR policy framework which is instead available for FD.io VPP implementation (Section V-B). This means that at the time of writing is not possible to create an SR policy (both MPLS and IPv6) and associate a *BindingSID* to it nor instantiate SR-MPLS/SRv6 steering rules pointing to SR-MPLS/SRv6 policies.

B. FD.io VPP

FD.io Vector Packet Processing (VPP) [206] platform is an extensible framework that provides out-of-the-box production quality switch/router functionality that can run on commodity CPUs. VPP 17.04 included the support for the *SRv6 Policy Headend* (formerly known as *transit*) behaviors and most of the *endpoint* behaviors defined in [3]. These behaviors are implemented in dedicated VPP graph nodes. The SRv6 graph nodes perform the required SRv6 behaviors as well the IPv6 processing (e.g., decrements Hop Limit). Whenever an SRv6 segment is instantiated, a new IPv6 FIB entry is created for the segment address pointing to the corresponding VPP graph node. Release 17.04 also brought SR headend capabilities to

VPP by introducing the concept of SR policy in the SRv6 implementation. In VPP, an SR policy is uniquely identified by its *BindingSID* address, which serves as a key to a particular SR policy. This is not compliant with the SR policy definition [33], but a reasonable shortcut considering the absence of control-plane capabilities in VPP.

The SR policies in VPP support several SID lists with weighted load-balancing of the traffic among them. When a new segment list is specified for an SR policy, VPP precomputes the rewrite string that will be used upon steering traffic into that SID list, either via an *SR Policy Headend* behavior or a *BindingSID*. VPP then initializes one FIB entry for the SR policy *BindingSID* in the FIB and an entry in a hidden FIB table for the IPv6 traffic steered into the SR policy via an *SR Policy Headend* behavior. Each one of these FIB entries points to the SR policy object, which in turn recurses on the weighted segment lists.

Traffic can be steered into an SR policy either by sending it to the corresponding *BindingSID* or by configuring a rule, called steering policy, that directs all traffic transiting towards a particular IP prefix or L2 interface into an SRv6 policy. The latter mechanism is implemented as FIB entry for the steered traffic in the main FIB to be resolved via the FIB entry of the SR policy in the hidden FIB table. In this way, a hierarchical FIB structure is realized: the traffic is not directly steered over an SR policy, but instead directed to a hidden FIB entry associated with the policy. This allows the SR policy to be modified without requiring any change to the steering rules that point towards it.

Release 17.04 has also seen the introduction of the SRv6 LocalSID development framework and the SR-MPLS implementation. The former is an API which allows developers to create new SRv6 *endpoint* behaviors using the VPP plugin framework. The principle is that the developer only codes the actual behavior, i.e., the VPP graph node. Instead, the segment instantiation, listing and removal are performed by the existing SRv6 code. The SR-MPLS framework is focused on the SR policies, as well on its steering. Likewise in SRv6, an SR policy is defined by a MPLS label representing the *BindingSID* and a weighted set of MPLS stacks of labels. Spray policies are a specific type of SR-MPLS policies where the packet is replicated on all the SID lists, rather than load-balanced among them. To steer packets in transit into an SR-MPLS policy, the user has to create an SR-MPLS steering policy. Instead, others SR-MPLS features, such as for example adjacency SIDs, can be achieved using the regular VPP MPLS implementation. In release 18.04, service programming proxy behaviors *End.AS*, *End.AD* and *End.AM* were introduced as VPP plugins leveraging the framework described before.

C. Other Open Source Implementations

Several research efforts analyzed in Section IV have released the components and the extensions realized for SR as open source. Some of them build upon the implementations described in the previous subsections, while other ones propose alternative solutions. The SREXT module ([163]) provides a complementary implementation of SRv6 in Linux

based nodes. When it was designed, the Linux kernel only offered the basic SRv6 processing (*End* behavior). SREXT complemented the SRv6 Linux kernel implementation providing a set of behaviors that were not supported yet. Currently most of the behaviors implemented in SREXT are supported by the mainline of Linux kernel (with the exception of the SR proxy behaviors). SREXT provides an additional local SID table which coexists with the one maintained by the Linux kernel. The SREXT module registers itself as a callback function in the *pre-routing* hook of the netfilter [200] framework. Since its position is at beginning of the netfilter processing, it is invoked for each received IPv6 packet. If the destination IPv6 address matches an entry in the local SID table, the associated behavior is applied otherwise the packet will follow the normal processing of the routing subsystem. The source code of SREXT together with the Vagrant box are available at [207]. Using the Vagrant box, it is possible to bootstrap a small testbed in few minutes and start the experiments on SREXT features.

FRRouting (FRR) [208] is an open source routing protocol stack for Linux forked from Quagga [209]. In FRR, there is an experimental support [210] of the draft [25] which defines the OSPFv2 extensions for Segment Routing (SR-MPLS). At the time of writing, there is no support for SRv6.

The SPRING-OPEN project [145] provides an SDN-based implementation of SR-MPLS. The architecture is based on a classic SDN control plane (logically centralized), built on top of ONOS. Part of this work converged later in Trellis [189], an open-source multi-purpose leaf-spine fabric supporting distributed access networks, NFV and edge cloud applications. Trellis includes also the support of P4/P4Runtime [211], [212] devices as well as Stratum [213] enabled devices. Trellis has been used as underlay/overlay fabric in the CORD project [214] which aims at redesigning central-office architectures. Recently, it has been integrated in the SEBA project [215] which targets residential-access networks. All the software stack and the documentation is freely available on [216]. Moreover, a tutorial together with a ready-to-go VM can be downloaded from [217].

PMSR ([15] and [114]) provides an open source implementation of SR-MPLS together with the realization of an SDN control plane. The data plane leverages the OSHI architecture ([218], [219]) which combines an SDN data plane, implemented with Open vSwitch [220], and OSPFv2 control logic, realized with Quagga. This architecture is extended in PMSR with the introduction of a Routes Extraction entity which connects to Quagga and receives routes update using the FPM interface provided by Quagga [209]. These routes are then translated in SIDs and installed in the SDN data plane as OpenFlow MPLS forwarding rules. Authors provide a set of management tools [221] which assist experimenters and relieve them from a huge configuration effort. A tutorial to start working with PMSR is available on [222]; instead a ready-to-go VM with all the dependencies installed can be downloaded from [223].

Software Resolved Network (SRN) (described in [148] and [149]) is a variant of the SDN architecture. The network controller is logically centralized and co-located with a DNS resolver and uses extensions of the DNS protocol to interact with end-hosts. The Open vSwitch Database Management Protocol (OVSDB) [224] is used to enable the communication between SDN controller and the network nodes: i) the latter populates the distributed database with the topology information and TE metadata; ii) the former once computed the path, upon a request, populates the OVSDB instance with the SRv6 Segment list matching the desired requirements. Finally, this is pulled by the access device which enables the communication of the end-hosts. The source code is freely available at [225]. An overview of the architecture can be found in [226]. A ready-to-go VM with packaged experiments can be created using the instructions in [227].

A classical SDN architecture for SRv6 technology is proposed in [4]: a centralized logic takes decisions on the Segment Lists that need to be applied to implement the services, then the SDN controller, using a southbound API, interacts with the SR enabled devices to enforce the application of such Segment Lists. The code related to the SDN architecture, i.e., the four different implementations of the Southbound API and the topology discovery, can be downloaded from the page of the project [228]. In addition the authors, to support both the development and testing aspects, have realized an Intent based emulation system to build realistic and reproducible experiments relieving the experimenters from a huge configuration effort. The emulation tools are available at [229].

SRV6Pipes [164] is an extension of the SRv6 implementation in the Linux kernel [193] which enables chaining and operation of in-network functions operating on streams. SRv6 policies are installed using the SRN architecture [148] described earlier in this section. However, SRN components are not mandatory since SRv6 policies can be installed in the edge nodes using the *iproute* utility. SRv6Pipes is composed by multiple components that necessarily need to be installed in the target machines: TCP proxy, patched Kernel and user space utilities. The minimum components can be downloaded from the repository of the project [230]. The complete code of the experiments together with a walkthrough can be found in [231].

SRNK [21] extends the implementation of SRv6 in the Linux kernel [193] adding the support for the *End.AS* and *End.AD* behaviors. The source code is freely available at [232], where it is possible to download the patched Linux kernel (starting from 4.14.0 branch) and the patched iproute2 (starting from iproute2-ss171112 tag). Instead, the detailed configurations steps of the SR-proxy are reported in [21, Appendices A and B].

The implementation of a path computation element able to compute robust disjoints SR paths, which remain disjoint even after an input set of failures without the need of configuration changes, is described in [133]. The java implementation of the algorithms, the public topologies used for the experiments, the experimental results and a detailed walkthrough to replicate the experiments of the paper are available at [233].

D. Hardware Implementations, Inter-Operability Efforts and Deployments for SRv6

An overview of IPv6 Segment Routing implementations is proposed in [234]. The work also describes some interoperability scenarios that have been demonstrated in public events and reports a list of recent SRv6 deployments in production networks.

With regards to the hardware implementations, eight vendors that declare production support of SRv6 for their hardware, are mentioned in [234]. The platforms ASR 1000, ASR 9000, NCS 5500, NCS 540 and NCS 560 are reported as the Cisco Routing platforms supporting SRH processing; ASR 9000 and NCS 5500 being deployed in production networks. As for Huawei, the reported platforms are: ATN, CX600, NE40E, ME60, NE5000E, NE9000 and NG-OLT MA5800, all with VRPV8 shipping code. The programmable devices based on Tofino chipset [235] can be programmed to support SRH processing. This is also true for the reference software implementation of the P4 devices [236], the Stratum based devices [213] and all the programmable chipsets (Cavium Xpliant [237] to give an example). Other SRv6 hardware implementations are reported for the Prestera family of Ethernet switches by Marvell, for the SkyFlux UAR500 router by UTStarcom. Spirent and Ixia also support SRv6, respectively in their TestCenter and IxNetwork testing platforms. Finally, as reported in [2], Juniper's Trio and vTrio NPUs have an experimental support of SRH (SRH insertion mode and *End* processing of interfaces addresses).

Three interoperability events are listed in [234]. The first (in chronological order) interoperability testing scenarios showcased at the 2017 SIGCOMM conference [238]. The set of experiments included a L3 VPN scenario augmented with TE functionality and services function chaining processing. SREXT, VPP, Linux kernel, Barefoot Tofino, Cisco NCS5500 and Cisco ASR1000 routers were the network devices implementing SRv6 behaviors. Iptables (firewall) and Snort (Intrusion Detection System) have been used as service functions. Finally, Wireshark and tcpdump have been leveraged to verify the proper operations of the network. The second set of interoperability test scenarios were run in March 2018 by the European Advanced Networking Test Center (EANTC) and their results [239] were presented at the MPLS + SDN + NFV World Congress conference in April 2018. In these tests, the implementations of CISCO and UTStarcom and the testing platforms of Ixia and Spirent were involved. The tests concerned Layer 3 IPv4 VPNs based on SRv6 (also including Traffic Engineering features in the SRv6 underlay) and SRH based Topology Independent (TI-LFA) Fast Reroute mechanisms. The third set of interoperability test scenarios were run in March 2019 and their results [239] were presented at the MPLS + SDN + NFV World Congress conference in April 2019. In these tests, a routing platform from Cisco (NCS 5500) and two from Huawei (NE9000-8 and NE40E-F1A) were involved. The tests concerned Layer 3 IPv4 and IPv6 VPNs based on SRV6, the validation of some SRv6 behaviors, SRv6 based fast reroute and OAM procedures (Ping and traceroute) based on [49].

Eight large scale deployments of SRv6 are listed in [234], involving the following nationwide operator networks: Softbank (Japan), China Telecom (China), Iliad (Italy), LINE Corporation (Japan), China Unicom (China), CERNET2 (China), MTN Uganda Ltd (Uganda), NOIA Network. We refer the reader to [234] for the details about these deployments.

The draft [234] elaborates also on the open source applications supporting the processing of the IPv6 Segment Routing header, among which we mention the well known Wireshark [240], tcpdump [241], iptables [242], nftables [243] and snort [244].

The implementation of a solution that offloads the SRH encapsulation/decapsulation operations and the SRv6 cache handling from the servers to the Intel FPGA programmable cards, is reported in [169]. Open Programmable Accelerator Engine [245] developed by Intel and the VPP software router are used as basic frameworks. The authors defined a new functional splitting of the SRv6 behaviors between the hardware and the software and the inter-working is realized introducing new graph nodes in VPP. These extensions to the VPP graph are able to encode/decode/process metadata exchanged between VPP and the Intel cards. For example, in the ingress direction, the card performs a SID lookup, and based on lookup result can strip the outer IPv6/SRH headers and add its own metadata header. Since VPP is augmented with new nodes is able to process this meta-header and the inner packets. As explained above, the solution partially offloads the functions to the hardware for example Route lookup and in general control plane functions are still done in software by VPP leveraging the CPU of the servers.

VI. Lessons Learned

In this section we describe the main outcomes learned during the survey activity. We first focus on some general considerations, then we elaborate on research related outcomes.

Despite the fact that the concept of Source Routing had already been proposed in the past (late 70âĂŹ), included in the initial specifications of IPv4 and implemented by some vendors, it was rarely deployed by network operators, mainly due to security issues. Novel SR architecture provides a secure solution for the deployment of the concept of source routing in operator's networks. SR also provide a scalable solution: i) the number of flow states to be maintained in network nodes is highly reduced, ii) only border nodes are involved by classification procedures and transit nodes do not maintain any flow state information.

SR-MPLS was the first deployment of SR concepts, only in the last few years SRv6 architecture has been defined. SR-MPLS architecture has, in our view, the main merit of not requiring any change to the MPLS forwarding plane. In this way, SR-MPLS represents a simple solution for the service provider with an IPv4 infrastructure. The same considerations apply to SRv6, the transition strategy from a classical IPv6 network to an SRv6 one can be realized in successive steps with no service disruption.

We have also noted that SR-MPLS implementation has not received the same attention as SRv6 is receiving from the research community. We believe that having available different open source implementations (Linux kernel and VPP) from the beginning has contributed decisively to this. Another reason is to be found in the required control plane extensions. Even though SR-MPLS has been the first implementation, the researchers had to struggle to have something to work with. MPLS data plane implementation has been unstable for a long time in the Linux kernel, and a completely revised implementation has been merged quite recently. Finally, SR-MPLS requires extensions to the routing protocols and there were no such extensions at the time that SR was initially being studied.

One of the main features of SR is the native support of NFV/SFC scenarios. Furthermore, the Network Programming model of SRv6 offers the possibility of virtualizing any service by combining the basic functions in a network program that is embedded in the packet header (implementing a network programming model on MPLS dataplane would not be scalable despite its feasibility). In this context, we believe that SRv6 can simplify network architectures with respect to SR-MPLS, since the use of different tunneling solutions can be avoided. As such, SRv6 is an attractive choice for operators that are deploying new networks or planning the evolution of their networking architectures.

During the revision of the Internet Drafts, RFCs and patents we noticed that there is a clear trend in the standardization activities to "follow" patents. In facts, most patents have also been standardized later or have been included in several standardization activities led by the same vendors. The most recent patents also highlight that SR can have a significant role in the deployment of networking core technologies, such as network slicing, 5G and cloud-based networking.

Regarding the outcomes of the research activity and following the classification provided in Section IV, we will briefly describe the most significant lessons coming from different research categories.

Research activity related to the *Monitoring* category (Section IV-A) highlights a significant interest towards solutions to improve existing monitoring tools or to define new tools exploiting SR features. Contrarily, there are lack of works focused on SR failure monitoring, such as misconfigurations, undefined SIDs, SR related black holes, packet loss, etc.

Moving to the *Traffic Engineering* category (Section IV-B), the main outcome of the reviewing activity is that one of the interesting features of SR routing, i.e., the capability of splitting the traffic over several SLs, is not exploited in practice. This unexpected outcome can be motivated by the capability of obtaining an high routing flexibility also with a single SL for each flow; in any case, the use of multiple SL per flow can be considered in the future to further improve TE solutions.

An additional consideration comes from the *Centrally Controlled Architecture* category (Section IV-D): the scientific community is oriented towards a centralized control plane for SR-MPLS networks, even if a distributed control plane is also supported. However, this approach requires extensions in the routing protocols in the case of MPLS, which represented a major barrier, thus causing many works to decide to leverage the openness of the SDN approach to overcome such limitations. It is also interesting to note that same trend cannot be found with works related to SRv6, where such extensions in routing protocols are not strictly required.

The *Path Encoding* (Section IV-E) related works highlight two interesting outcomes. As first, the large effort taken by researchers in reducing the overhead in the packet headers demonstrates that is seen as a limiting factor for SR deployments. Two schools of thought can be found: i) optimize the translation of the paths into SLs; ii) reduce its impact using a smarter encoding of the paths into segment lists.

The second lesson learned is related to the adj-sids; they are largely used by path encoding works to increase routing flexibility, while they are not fully exploited in TE works: the reason could be that including adj-sids leads to an increase in complexity for optimization models used by TE algorithms. It will be interesting to investigate the impact of adj-sids in novel SR-based TE solutions both in terms of performance and complexity.

The works related to the *Network Programming* category (Section IV-F) suggest that this SRv6 feature opens up a wide range of research opportunities: it can be applied in different use cases, from Service Function Chaining to the implementation of complex operations (such as the management of VM migration in a DC network). We believe that network programming will attract significant attention in the coming years.

Finally, an interesting consideration of the *Performance Evaluation* category (Section IV-G) is related to the topics of the analyzed works. All papers concentrate on the data plane, while the only work studying control plane performance is [4] (actually only a performance evaluation of different southbound implementations has been conducted).

VII. FUTURE RESEARCH DIRECTIONS

Most of the SR works we have reviewed have focused on the definition of novel solutions for classical network problems (such as Monitoring, Traffic Engineering and Failure Recovery) or on optimization of specific SR procedures (such as Path Encoding). In general, these works showed that SR can provide significant enhancements with respect to other solutions and we believe that there is still room and interest for extending the achieved results in these areas. In addition, we try to identify and discuss a set of research directions for Segment Routing that are definitely worth exploring in the near future: i) Service Function Chaining support, ii) SRv6 end-host implementation aspects, iii) Cloud Orchestration, iv) Integration with Applications, v) 5G, vi.) IoT. All these research areas are based on SRv6, i.e., on Segment Routing over the IPv6 data plane, as we believe that the future evolution of Segment Routing will be based on SRv6.

A. Service Function Chaining Support

The Programmability feature of SRv6 represents an enabling factor for the implementation of Network Function Virtualization (NFV) and Service Function Chaining (SFC) in provider networks. In this regard, new abstraction models for the management of Network Functions by means of dedicated SRv6 control procedures could be studied.

TABLE AI
TABLE OF THE ABBREVIATIONS

Acronym	Extended
Adj-SID	Adjacency SID
API	Application Programming Interface
ASIC	application specific integrated circuit
ASR	Aggregation Services Router
BFD	Bidirectional Forwarding Detection
BGP	Border Gateway Protocol
BGP-LS	BGP Link State
BSID	Binding Segment IDentifier
CCA	Centrally Controlled Architectures
CDN	Content Delivery Network
CLI	Command Line Interface
CE	Carrier Ethernet
CP	Constrained Programming
CSP	Cloud Service Provider
CXP	Control Exchange Poin
CORD	Central Office Re-architected as Datacenter
DAG	Direct Acyclic Graph
DC	Data Center
DEFO	Declarative and Expressive Forwarding Optimizer
DEFT	Distributed Exponentially-weighted Flow SpliTting
Diffserv	Differentiated Services
DNS	Domain Name System
DSL	Domain Specific Language
EANTC	European Advanced Networking Test Center
eBPF	extended Berkeley Packet Filter
ECMP	Equal Cost MultiPath
ERO	Explicit Routing Object
FEC	Forward Equivalent Class
FG	Forwarding Graph
FIB	Forwarding Information Base
Flex-Algo	SR Flexible Algorithm
FUNCT	FUNCTion
FPM	Forwarding Plane Manager
FR	Failure Recovery
FRR	Fast ReRoute
GMPLS	Generalized MPLS
GRE	Generic Routing Encapsulation
gRPC	Remote Procedure Calls
GTP	GPRS Tunnelling Protocol
HMAC	Hashed Message Authentication Code
IE	Ingress-Egress
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
ILP	Integer Linear Programming
IP	Internet Protocol
IPv6	IP version 6
ISIS	Intermediate System to Intermediate System
ISP	Internet Service Provider
LAN	Local Area Network
LB	Load Balancer
LDP	Label Distribution Protocol
LER	Label Edge Router
LOC	LOCator
LRU	Least-Recently-Used
LS	Local Search
LSP	Label Switched Path
LSR	Label Switching Router
LSRR	Loose Source and Record Route
LWT	Linux Lightweight Tunnel
M-SRD	Multiple Segment Routing Domain
MILP	Mixed Integer Linear Programming
MIRA	Minimum Interference Routing Algorithm
MISC	Miscellaneous
MON	Monitoring
MOPSO	Multiple Objective Particle Swarm Optimization
MPLS	Multi Protocol Label Switching
MPLS-TE	MPLS Traffic Engineering
MPTCP	MultiPath TCP
MR	Middle-point Routing
MSD	Maximum Stack Depth
M-SRD	Multiple Segment Routing Domain
NAT	Network Address Translation
NB	North Bound
NCS	Network Convergence System
NDR	No-Drop Rate
NETCONF	Network Configuration Protocol
NFV	Network Function Virtualization

Acronym	Extended
NIC	Network Interface Card
NP	Network Programming
NSC	Network Service Chaining
NSH	Network Service Header
NSO	Network Service Orchestrator
OAM	Operation and Maintenance
OD	Origin-Destination
ODL	OpenDayLight
OE	Omnipresent Ethernet
OF	OpenFlow
OF-DPA	OpenFlow Data Plane Abstraction
OFSL	OpenFlow Switching Layer
OSHI	Open Source Hybrid IP/SDN
OSPF	Open Shortest Path First
OTT	Over The Top
OVSDB	Open vSwitch Database
PaCoB	Path Cost Based heuristic
PCC	Path Computation Client
PCE	Path Computation Element
PCEP	Path Computation Element Communication Protocol
PDR	Partial Drop Rate
PE	Path Encoding
PEFT	Penalizing Exponential Flow-spliTting
PMSR	Poor Man's Segment Routing
QoE	Quality of Experience
QoMRA	QoE-Centric Multipath Routing Algorithm
QoS	Quality of Service
REST	Representational State Transfer
RFC	Request For Comment
RO	Resource Orchestrator
ROADMS	Reconfigurable Optical Add-Drop Multiplexer
S-SRD	Single Segment Routing Domain
SD-WAN	Software Defined WAN
SDN	Software Defined Networking
SEBA	SDN Enabled Broadband Access
SERA	SR Aware
SERPENT	Segment Routing Perturbation Traffic
SFC	Service Function Chaining
SID	Segment IDentifier
SL	Segment List
SLA	Service Level Agreement
SLD	Segment List Depth
SDN	Software Defined Networking
SPF	Shortest Path First
SPRING	Source Packet Routing In Networking
SR	Segment Routing
SRD	SR Domain
SREXT	SR EXTensions
SR-LEA	SR Label Encoding Algorithm
SR-MPLS	SR over MPLS
SRGB	Segment Routing Global Block
SRH	Segment Routing Header
SRLB	Segment Routing Load Balancing
SRLG	Shared Risk Link Group
SRN	Software Resolved Networks
SRNK	SR-Proxy Native Kernel
SRCBA	SFC Routing and Cloud Bandwidth resource Allocation
SRv6	SR over IPv6
SSRR	Strict Source and Record Route
SWP	Shortest Widest Path
TCAM	Ternary Content Addressable Memory
TCP	Transport Control Protocol
TE	Traffic Engineering
TI-LFA	Topology Independent LoopFree Alternate
TI-MFA	Topology Independent Multi-Failure Alternate
TLV	Type Length Value
TM	Traffic Matrix
TSID	Targeted SID
UDP	User Datagram Protocol
VLAN	Virtual LAN
VM	Virtual Machine
VNF	Virtual Network Function
VPN	Virtual Private Network
VPP	Vector Packet Processing
VXLAN	Virtual Extensible Local Area Network
WAN	Wide Area Network
WG	Working Group
WSP	Wide Shortest Path

B. SRv6 End-Host Implementation Aspects

Another interesting topic is related to the implementation of SRv6 in end-hosts. One aspect is related to moving SRv6 functions in end-hosts from the software closer to the hardware with SmartNICs. Programmable NICs allow to implement network traffic processing on the NIC instead of using the CPU of the end-nodes/devices. Another aspect is related to exploiting the recent advances in Linux kernel networking for fast packet processing, namely eBPF [185] and XDP [246] for implementing SRv6 functions.

C. Cloud Orchestration

The third research opportunity regards integration of the SRv6 technology into Cloud orchestrators like OpenStack [247] and Kubernetes [248]. Considering Data Center networking scenarios, it will be possible to replace actual data plane mechanism based on legacy tunneling mechanisms like VXLAN with SRv6, with a drastic simplification of the network stack: the needed information will be integrated into SRv6 SIDs and/or in the TLV field, with no need of dedicated headers for tunneling.

D. Integration With Applications

Allowing direct interaction of applications with SRv6 features could enable innovative services and improve the efficiency of existing ones. Applications could use SRv6 SIDs to express their service requirements and to interact with network features, dynamically participating in the definition/composition of network services. To achieve this interaction, first the APIs of the operating system (e.g., Linux) need to be extended, then SRv6 aware applications need to be developed.

E. 5G

SRv6 is being considered for the data plane of future releases of 5G networks thanks to its stateless traffic steering and programmability features. On one hand, SRv6 could support 5G features like network slicing, on the other hand, it will be important to evaluate the performance of SRv6 based data plane, to verify that strict 5G constraints on latency are met.

F. Internet of Things

Considering the problem space of Internet of Things, which includes scalability aspects, routing aspects, interactions between networking and application layers, the application of the SRv6 architecture seems very promising.

VIII. Conclusion

Segment Routing technology is based on source routing and tunneling paradigms. Segment Routing supports services such as Traffic Engineering, Virtual Private Networks, Fast Restoration in IP backbones and datacenters, and has proved to be flexible in supporting new use cases. Moreover, SR architecture reduces the amount of state information that needs to be configured in the core nodes.

SR-MPLS and SRv6 are the two data plane instantiations of SR architecture. This is the first tutorial and survey work covering in detail the novel SRv6 solution (i.e., SR over IPv6 data plane), which represents the most promising implementation for future research activity. SRv6 provides a consistent solution for solving long-term problems in IP networks, simplifying protocol stacks and improving scalability with respect to current solutions.

In the survey we covered standardization work, patents and research activities related to Segment Routing. We also considered the recent deployments of SR in real networks and existing SR implementations, with a focus on the open source tools that can support SR research and development activities.

As for research activities, we covered about 90 scientific papers related to SR and proposed a taxonomy for their classification. One of the main outcomes of the classification was to identify relationships between SR features and research topics. For instance, the source routing paradigm has turned out to be the key enabling feature for the implementation of Traffic Engineering solutions in an SR network, while the routing flexibility feature is mainly used to realize network monitoring tools. We also identified the most interesting SR standardization documents and provided a taxonomy for their classification. A number of patents related to Segment Routing has also been discussed.

The review of SR implementations has highlighted the maturity of open source solutions based on Linux kernel and VPP. Both Linux and VPP, widely used by the research and developer community, allow the easy deployment of a virtual SR playground. As part of our survey activity we have also reported our vision and our experience in terms of lessons learned and future research topics.

We hope that this tutorial and survey work will draw further attention from the research community to Segment Routing technology and motivate new researchers to join the development of new use cases and standardization efforts. We have anticipated future research directions which can be taken as starting points.

New versions of this survey will be available at [249]. Moreover, we strongly encourage the community to provide feedback and updates as new research works and SR deployments come out and technology evolves. For this reason we have created a public repository,[2] where interested researchers can contribute and update this documentation.

Appendix A

See Table A1.

References

[1] A. Bashandy, C. Filsfils, S. Previdi, B. Decraene, S. Litkowski, and R. Shakir, "Segment routing with the MPLS data plane," IETF, RFC 8660, Dec. 2019. [Online]. Available: https://tools.ietf.org/html/rfc8660

[2] C. Filsfils, D. Dukes, S. Previdi, J. Leddy, S. Matsushima, and D. Voyer, "IPv6 segment routing header (SRH)," IETF, RFC 8754, Mar. 2020. [Online]. Available: https://tools.ietf.org/html/rfc8754

[3] C. Filsfils, P. Camarillo, J. Leddy, D. Voyer, S. Matsushima, and Z. Li, "SRv6 network programming," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-spring-srv6-network-programming, Jan. 2020. [Online]. Available: https://tools.ietf.org/html/draft-ietf-spring-srv6-network-programming

[4] P. L. Ventre, M. M. Tajiki, S. Salsano, and C. Filsfils, "SDN architecture and southbound APIs for IPv6 segment routing enabled wide area networks," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 4, pp. 1378–1392, Dec. 2018.

[5] J. Postel, Ed., "Internet protocol," IETF, RFC 791, Sep. 1981. [Online]. Available: https://tools.ietf.org/html/rfc791

[6] A. Farrel, O. Komolafe, and S. Yasukawa, "An analysis of scaling issues in MPLS-TE core networks," IETF, RFC 5439, Feb. 2009. [Online]. Available: https://tools.ietf.org/html/rfc5439

[7] S. Previdi, C. Filsfils, B. Decraene, S. Litkowski, M. Horneffer, and R. Shakir, "Source packet routing in networking (SPRING) problem statement and requirements," IETF, RFC 7855, May 2016. [Online]. Available: https://tools.ietf.org/html/rfc7855

[2]https://github.com/netgroup/sr-survey

[8] C. Filsfils, S. Previdi, B. Decraene, and R. Shakir, "Resiliency use cases in source packet routing in networking (SPRING) networks," IETF, RFC 8355, Mar. 2018. [Online]. Available: https://tools.ietf.org/html/rfc8355

[9] J. Brzozowski, J. Leddy, C. Filsfils, R. Maglione, and M. Townsley, "Use cases for IPv6 source packet routing in networking (SPRING)," IETF, RFC 8354, Mar. 2018. [Online]. Available: https://tools.ietf.org/html/rfc8354

[10] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment routing architecture," IETF, RFC 8402, Jul. 2018. [Online]. Available: https://tools.ietf.org/html/rfc8402/

[11] R. Geib, C. Filsfils, C. Pignataro, and N. Kumar, "A scalable and topology-aware MPLS data-plane monitoring system," IETF, RFC 8403, Jul. 2018. [Online]. Available: https://tools.ietf.org/html/rfc8403

[12] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. François, "The segment routing architecture," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2015, pp. 1–6.

[13] A. Farrel and R. Bonica, "Segment routing: Cutting through the hype and finding the IETF's innovative nugget of gold," *IETF J.*, vol. 13, no. 1, pp. 5–9, Jul. 2017. [Online]. Available: https://www.ietfjournal.org/segment-routing-cutting-through-the-hype-and-finding-the-ietfs-innovative-nugget-of-gold/

[14] Z. N. Abdullah, I. Ahmad, and I. Hussain, "Segment routing in software defined networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 464–486, 1st Quart., 2019.

[15] S. Salsano, L. Veltri, L. Davoli, P. L. Ventre, and G. Siracusano, "PMSR-poor man's segment routing, a minimalistic approach to segment routing and a traffic engineering use case," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, 2016, pp. 598–604.

[16] R. Bonica, C. Pignataro, and J. Touch, "Generic routing encapsulation (GRE)," IETF, RFC 2784, Mar. 2000. [Online]. Available: https://tools.ietf.org/html/rfc2784

[17] M. Mahalingam *et al.*, "Virtual extensible local area network (VXLAN): A framework for overlaying virtualized layer 2 networks over layer 3 networks," IETF, RFC 7348, Aug. 2014. [Online]. Available: https://tools.ietf.org/html/rfc7348

[18] H. Ichihara and T. Tsuchiya. *LINE Datacenter Networking With SRv6*. Accessed: Dec. 3, 2020. [Online]. Available: https://speakerdeck.com/line_developers/line-data-center-networking-with-srv6

[19] R. Browne, A. Chilikin, and T. Mizrahi, "Network service header (NSH)," IETF, RFC 8300, Jan. 2018. [Online]. Available: https://tools.ietf.org/html/rfc8300

[20] *Does 5G Mean We Can Finally Update the Mobile Data Plane Protocol*. Accessed: Dec. 3, 2020. [Online]. Available: https://www.metaswitch.com/blog/does-5g-mean-we-can-finally-update-the-mobile-data-plane-protocol

[21] A. Mayer *et al.*, "An efficient linux kernel implementation of service function chaining for legacy VNFs based on IPv6 segment routing," in *Proc. 5th IEEE Int. Conf. Netw. Softw. (NetSoft)*, 2019, pp. 333–341.

[22] C. Filsfils *et al.*, "Indexed segment ID," U.S. Patent 9 559 954 B2, 2014.

[23] F. Clad *et al.*, "Service programming with segment routing," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-spring-sr-service-programming-01, Oct. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-spring-sr-service-programming

[24] S. Matsushima *et al.*, "Segment routing IPv6 for mobile user plane," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-dmm-srv6-mobile-uplane, Mar. 2019. [Online]. Available: https://tools.ietf.org/html/draft-ietf-dmm-srv6-mobile-uplane

[25] P. Psenak *et al.*, "OSPF extensions for segment routing," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-ospf-segment-routing-extensions, Dec. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-ospf-segment-routing-extensions

[26] P. Psenak and S. Previdi, "OSPFv3 extensions for segment routing," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-ospf-ospfv3-segment-routing-extensions, Jan. 2019. [Online]. Available: https://tools.ietf.org/html/draft-ietf-ospf-ospfv3-segment-routing-extensions

[27] S. Previdi, L. Ginsberg, C. Filsfils, A. Bashandy, H. Gredler, and B. Decraene, "IS-IS extensions for segment routing," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-isis-segment-routing-extensions, May 2019. [Online]. Available: https://tools.ietf.org/html/draft-ietf-isis-segment-routing-extensions

[28] P. Psenak, C. Filsfils, A. Bashandy, B. Decraene, and Z. Hu, "IS-IS extensions to support routing over IPv6 dataplane," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-lsr-isis-srv6-extensions, May 2019. [Online]. Available: https://tools.ietf.org/html/draft-ietf-lsr-isis-srv6-extensions

[29] Z. Li, Z. Hu, D. Cheng, K. Talaulikar, and P. Psenak, "OSPFv3 Extensions for SRv6," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-li-ospf-ospfv3-srv6-extensions, Mar. 2019. [Online]. Available: https://tools.ietf.org/html/draft-li-ospf-ospfv3-srv6-extensions

[30] C. Filsfils, T. F. Telkamp, and A. T. Gous, "Segment routing based wide area network orchestration in a network environment," U.S. Patent 9 647 944 B2, 2014.

[31] H. Gredler, "BGP link-state extensions for segment routing," U.S. Patent 9 660 897 B1, 2014.

[32] S. Sivabalan, C. Filsfils, J. Tantsura, W. Henderickx, and J. Hardwick, "PCEP extensions for segment routing," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-pce-segment-routing, Oct. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-pce-segment-routing

[33] C. Filsfils, K. Talaulikar, P. Król, M. Horneffer, and P. Mattes, "Segment routing policy architecture," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-spring-segment-routing-policy, Oct. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-spring-segment-routing-policy

[34] C. Filsfils *et al.*, "SR policy implementation and deployment considerations," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-filsfils-spring-sr-policy-considerations, Oct. 2018. [Online]. Available: https://tools.ietf.org/html/draft-filsfils-spring-sr-policy-considerations

[35] K. Raza *et al.*, "YANG data model for segment routing policy," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-raza-spring-sr-policy-yang, May 2019. [Online]. Available: https://tools.ietf.org/html/draft-raza-spring-sr-policy-yang

[36] C. Filsfils *et al.*, "Illustrations for SRv6 network programming," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-filsfils-spring-srv6-net-pgm-illustration, Feb. 2019. [Online]. Available: https://tools.ietf.org/html/draft-filsfils-spring-srv6-net-pgm-illustration

[37] C. Filsfils, S. Previdi, G. Dawra, E. Aries, and P. Lapukhov, "BGP-prefix segment in large-scale data centers," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-spring-segment-routing-msdc, Nov. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-spring-segment-routing-msdc

[38] C. Filsfils, S. Previdi, G. Dawra, E. Aries, and D. Afanasiev, "Segment routing centralized BGP peer engineering," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-spring-segment-routing-central-epe, Dec. 2017. [Online]. Available: https://tools.ietf.org/html/draft-ietf-spring-segment-routing-central-epe

[39] C. Filsfils, S. Previdi, G. Dawra, W. Henderickx, and D. Cooper, "Interconnecting millions of endpoints with segment routing," IETF, RFC 8604, Jun. 2019. [Online]. Available: https://rfc-editor.org/doc/rfc8604

[40] D. Dukes *et al.*, "SR for SDWAN—VPN with underlay SLA," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-dukes-spring-sr-for-sdwan, Dec. 2018. [Online]. Available: https://tools.ietf.org/html/draft-dukes-spring-sr-for-sdwan

[41] P. Camarillo, C. Filsfils, L. Bertz, A. Akhavain, S. Matsushima, and D. Voyer, "Segment routing IPv6 for mobile user-plane PoCs," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-camarillo-dmm-srv6-mobile-pocs, Oct. 2018. [Online]. Available: https://tools.ietf.org/html/draft-camarillo-dmm-srv6-mobile-pocs

[42] Z. Ali, C. Filsfils, P. Camarillo, and D. Voyer, "Building blocks for slicing in segment routing network," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ali-spring-network-slicing-building-blocks, Mar. 2019. [Online]. Available: https://tools.ietf.org/html/draft-ali-spring-network-slicing-building-blocks

[43] C. Filsfils, Z. Ali, M. Horneffer, D. Voyer, M. Durrani, and R. Raszuk, "Segment routing traffic accounting counters," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-filsfils-spring-sr-traffic-counters, Jun. 2018. [Online]. Available: https://tools.ietf.org/html/draft-filsfils-spring-sr-traffic-counters

[44] M. Anand *et al.*, "Packet-optical integration in segment routing," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-anand-spring-poi-sr, Jan. 2019. [Online]. Available: https://tools.ietf.org/html/draft-anand-spring-poi-sr

[45] S. Litkowski, A. Bashandy, C. Filsfils, B. Decraene, and B. Voyer, "Topology independent fast reroute using segment routing," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-rtgwg-segment-routing-ti-lfa, Mar. 2019. [Online]. Available: https://tools.ietf.org/html/draft-ietf-rtgwg-segment-routing-ti-lfa

[46] A. Bashandy, C. Filsfils, S. Litkowski, B. Decraene, P. Francois, and P. Psenak, "Loop avoidance using segment routing," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-bashandy-rtgwg-segment-routing-uloop, Sep. 2018. [Online]. Available: https://tools.ietf.org/html/draft-bashandy-rtgwg-segment-routing-uloop

[47] S. Hegde and P. Sarkar, "Micro-loop avoidance using SPRING," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-hegde-rtgwg-microloop-avoidance-using-spring, Jul. 2017. [Online]. Available: https://tools.ietf.org/html/draft-hegde-rtgwg-microloop-avoidance-using-spring/

[48] N. Kumar, C. Pignataro, G. Swallow, N. Akiya, S. Kini, and M. G. Chen, "Label switched path (LSP) ping/traceroute for segment routing (SR) IGP-prefix and IGP-adjacency segment identifiers (SIDs) with MPLS data planes," IETF, RFC 8287, Dec. 2017. [Online]. Available: https://tools.ietf.org/html/rfc8287

[49] Z. Ali, C. Filsfils, S. Matsushima, D. Voyer, and M. Chen, "Operations, administration, and maintenance (OAM) in segment routing networks with IPv6 data plane (SRv6)," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-6man-spring-srv6-oam-03, Dec. 2019. [Online]. Available: https://tools.ietf.org/html/draft-ietf-6man-spring-srv6-oam-03

[50] Z. Ali *et al.*, "Traffic accounting in segment routing networks," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ali-spring-sr-traffic-accounting, Jun. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ali-spring-sr-traffic-accounting

[51] Z. Ali, K. Talaulikar, C. Filsfils, N. Nainar, and C. Pignataro, "Bidirectional forwarding detection (BFD) for segment routing policies for traffic engineering," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ali-spring-bfd-sr-policy, Oct. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ali-spring-bfd-sr-policy

[52] R. Gandhi, C. Filsfils, D. Voyer, S. Salsano, and M. Chen, "In-band performance measurement for segment routing networks with MPLS data plane," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-gandhi-spring-rfc6374-srpm-mpls, Feb. 2019. [Online]. Available: https://tools.ietf.org/html/draft-gandhi-spring-rfc6374-srpm-mpls

[53] R. Gandhi, C. Filsfils, D. Voyer, S. Salsano, and M. Chen, "In-band performance measurement using UDP path for segment routing networks," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-gandhi-spring-rfc6374-srpm-udp, Feb. 2019. [Online]. Available: https://tools.ietf.org/html/draft-gandhi-spring-rfc6374-srpm-udp

[54] R. Gandhi, C. Filsfils, D. Voyer, M. Chen, and B. Janssens, "Performance measurement using TWAMP light for segment routing networks," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-gandhi-spring-twamp-srpm, Dec. 2019. [Online]. Available: https://tools.ietf.org/html/draft-gandhi-spring-twamp-srpm

[55] D. Frost and S. Bryant, "Packet loss and delay measurement for MPLS networks," IETF, RFC 6374, Sep. 2011. [Online]. Available: https://tools.ietf.org/html/rfc6374

[56] S. Bryant, S. Sivabalan, and S. Soni, "UDP return path for packet loss and delay measurement for MPLS networks," IETF, RFC 7876, Jul. 2016. [Online]. Available: https://tools.ietf.org/html/rfc7876

[57] A. Bashandy, C. Filsfils, S. Previdi, B. Decraene, and S. Litkowski, "Segment routing interworking with LDP," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-spring-segment-routing-ldp-interop, Sep. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-spring-segment-routing-ldp-interop

[58] P. Sarkar, H. Gredler, C. Filsfils, S. Previdi, B. Decraene, and M. Horneffer, "Anycast segments in MPLS based segment routing," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-spring-mpls-anycast-segments, Jan. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-spring-mpls-anycast-segments

[59] C. Filsfils, S. Previdi, P. Psenak, and L. Ginsberg, "Segment routing recursive information," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-filsfils-spring-sr-recursing-info, Jun. 2017. [Online]. Available: https://tools.ietf.org/html/draft-filsfils-spring-sr-recursing-info

[60] H. Sitaraman, V. Beeram, I. Minei, and S. Sivabalan, "Recommendations for RSVP-TE and segment routing (SR) label switched path (LSP) coexistence," IETF, RFC 8426, Jul. 2018. [Online]. Available: https://tools.ietf.org/html/rfc8426

[61] X. Xu, S. Bryant, A. Farrel, S. Hassan, W. Henderickx, and Z. Li, "SR-MPLS over IP," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-mpls-sr-over-ip, Jun. 2019. [Online]. Available: https://tools.ietf.org/html/draft-ietf-mpls-sr-over-ip

[62] D. Voyer *et al.*, "Insertion of IPv6 segment routing headers in a controlled domain," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-voyer-6man-extension-header-insertion, Jun. 2018. [Online]. Available: https://tools.ietf.org/html/draft-voyer-6man-extension-header-insertion

[63] K. Raza *et al.*, "YANG data model for SRv6 base and static," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-raza-spring-srv6-yang, Mar. 2018. [Online]. Available: https://tools.ietf.org/html/draft-raza-spring-srv6-yang

[64] S. Previdi, C. Filsfils, A. Lindem, A. Sreekantiah, and H. Gredler, "Segment routing prefix SID extensions for BGP," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-idr-bgp-prefix-sid, Jun. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-idr-bgp-prefix-sid-27

[65] S. Previdi *et al.*, "Advertising segment routing policies in BGP," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-idr-segment-routing-te-policy, Jul. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-idr-segment-routing-te-policy

[66] G. Dawra *et al.*, "SRv6 BGP based overlay services," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-dawra-bess-srv6-services, Mar. 2019. [Online]. Available: https://tools.ietf.org/html/draft-dawra-bess-srv6-services

[67] S. Previdi, K. Talaulikar, C. Filsfils, H. Gredler, and M. Chen, "BGP link-state extensions for segment routing," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-idr-bgp-ls-segment-routing-ext, May 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-idr-bgp-ls-segment-routing-ext

[68] S. Previdi, K. Talaulikar, J. Dong, M. Chen, H. Gredler, and J. Tantsura, "Distribution of traffic engineering (TE) policies and state using BGP-LS," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-idr-te-lsp-distribution, Jun. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-idr-te-lsp-distribution

[69] S. Previdi, K. Talaulikar, C. Filsfils, K. Patel, S. Ray, and J. Dong, "BGP-LS extensions for segment routing BGP egress peer engineering," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-idr-bgpls-segment-routing-epe, Mar. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-idr-bgpls-segment-routing-epe

[70] J. Tantsura, U. Chunduri, K. Talaulikar, G. Mirsky, and N. Triantafillis, "Signaling maximum SID depth using border gateway protocol link-state," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-idr-bgp-ls-segment-routing-msd, Oct. 2017. [Online]. Available: https://tools.ietf.org/html/draft-ietf-idr-bgp-ls-segment-routing-msd

[71] L. Ginsberg, S. Previdi, Q. Wu, J. Tantsura, and C. Filsfils, "BGP—Link state (BGP-LS) advertisement of IGP traffic engineering performance metric extensions," IETF, RFC 8571, Mar. 2019. [Online]. Available: https://tools.ietf.org/html/rfc8571

[72] G. Dawra *et al.*, "BGP link state extensions for IPv6 segment routing (SRv6)," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-idr-bgpls-srv6-ext, Mar. 2018. [Online]. Available: https://tools.ietf.org/html/draft-dawra-idr-bgpls-srv6-ext

[73] K. Talaulikar, C. Filsfils, K. Swamy, S. Zandi, G. Dawra, and M. Durrani, "BGP link-state extensions for BGP-only fabric," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ketant-idr-bgp-ls-bgp-only-fabric, Mar. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ketant-idr-bgp-ls-bgp-only-fabric

[74] G. Dawra *et al.*, "BGP-LS advertisement of segment routing service segments," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-dawra-idr-bgp-ls-sr-service-segments, Jan. 2019. [Online]. Available: https://tools.ietf.org/html/draft-dawra-idr-bgp-ls-sr-service-segments

[75] P. Psenak, S. Hegde, C. Filsfils, K. Talaulikar, and A. Gulko, "IGP flexible algorithm," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-lsr-flex-algo, Nov. 2018. [Online]. Available: https://tools.ietf.org/html/draft-ietf-lsr-flex-algo

[76] M. Rasool, S. Sreenivasan, and M. Negi, "YANG model for IGP flexible algorithm," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-rasool-lsr-flex-algo-yang, Apr. 2019. [Online]. Available: https://tools.ietf.org/html/draft-rasool-lsr-flex-algo-yang

[77] J. Tantsura, U. Chunduri, S. Aldrin, and L. Ginsberg, "Signaling maximum SID depth (MSD) using IS-IS," IETF, RFC 8491, Nov. 2018. [Online]. Available: https://tools.ietf.org/html/rfc8491

[78] L. Ginsberg, A. Bashandy, C. Filsfils, M. Nanduri, and E. Aries, "Advertising L2 bundle member link attributes in IS-IS," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-ietf-isis-l2bundles, May 2017. [Online]. Available: https://tools.ietf.org/html/draft-ietf-isis-l2bundles

[79] L. Ginsberg, S. Previdi, S. Giacalone, D. Ward, J. Drake, and Q. Wu, "IS-IS traffic engineering (TE) metric extensions," IETF, RFC 7810, May 2016. [Online]. Available: https://tools.ietf.org/html/rfc7810

[80] J. Tantsura, U. Chunduri, S. Aldrin, and P. Psenak, "Signaling maximum SID depth (MSD) using OSPF," IETF, RFC 8476, Dec. 2018. [Online]. Available: https://tools.ietf.org/html/rfc8476

[81] S. Giacalone, D. Ward, J. Drake, A. Atlas, and S. Previdi, "OSPF traffic engineering (TE) metric extensions," IETF, RFC 7471, Mar. 2015. [Online]. Available: https://tools.ietf.org/html/rfc7471

[82] S. Sivabalan, C. Filsfils, J. Tantsura, J. Hardwick, S. Previdi, adn C. Li, "Carrying binding label/segment-ID in PCE-based networks," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-sivabalan-pce-binding-label-sid, Feb. 2019. [Online]. Available: https://tools.ietf.org/html/draft-sivabalan-pce-binding-label-sid

[83] A. Rodriguez-Natal, V. Ermagan, F. Maino, D. Dukes, P. Camarillo, and C. Filsfils, "LISP control plane for SRv6 endpoint mobility," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-rodrigueznatal-lisp-srv6, Jan. 2019. [Online]. Available: https://tools.ietf.org/html/draft-rodrigueznatal-lisp-srv6

[84] C. Filsfils *et al.*, "Segment routing techniques," U.S. Patent 9 929 946 B2, 2013.

[85] S. Previdi and C. Filsfils, "Segment routing extension headers," U.S. Patent 10 063 475 B2, 2014.

[86] D. Frost *et al.*, "MPLS segment-routing," U.S. Patent 9 049 233 B2, 2013.

[87] C. Filsfils *et al.*, "Seamless segment routing," U.S. Patent 9 450 829 B2, 2013.

[88] C. Filsfils *et al.*, "Creating and maintaining segment routed traffic engineering policies via border gateway protocol," U.S. Patent 10 454 821 B2, 2017.

[89] S. Previdi *et al.*, "Advertisement of adjacency segment identifiers," U.S. Patent 9 565 160 B2, 2014.

[90] C. Filsfils *et al.*, "Method and system for path monitoring using segment routing," U.S. Patent 9 369 371 B2, 2015.

[91] C. Filsfils *et al.*, "Segment routing into a label distribution protocol domain," U.S. Patent 9 749 187 B2, 2014.

[92] T. Saad *et al.*, "Loop detection and avoidance for segment routed traffic engineered paths," U.S. Patent 10 182 000 B2, 2016.

[93] P. Francois *et al.*, "Loop avoidance during network convergence in switched networks," U.S. Patent 9 401 858 B2, 2014.

[94] S. Previdi and C. Filsfils, "Fast reroute for segment routing traffic," U.S. Patent 9 485 150 B2, 2016.

[95] C. Filsfils *et al.*, "Encoding explicit paths as segment routing segment lists," U.S. Patent 9 979 601 B2, 2018.

[96] C. Filsfils *et al.*, "Scalable network slice based queuing using segment routing flexible algorithm," U.S. Patent 10 601 724 B1, 2020.

[97] S. A. Dodd-Noble *et al.*, "Providing user equipment location information indication on user plane," U.S. Patent 10 285 155 B1, 2019.

[98] Y. Desmouceaux *et al.*, "Zero-loss workload mobility with segment routing for virtual machines," U.S. Patent 10 635 480 B2, 2019.

[99] *GTP-U Specification*. Accessed: Dec. 3, 2020. [Online]. Available: https://www.3gpp.org/dynareport/29281.htm

[100] M. T. Y. Desmouceaux and T. Clausen, "Zero-loss virtual machine migration with IPv6 segment routing," in *Proc. 1st Workshop Segment Routing Service Function Chaining (SR+SFC) CNSM*, Rome, Italy, 2018, pp. 420–425.

[101] F. Aubry, D. Lebrun, S. Vissicchio, M. T. Khong, Y. Deville, and O. Bonaventure, "Interoperable multi-domain delay-aware provisioning using Segment Routing monitoring and BGP-LS advertisement," in *Proc. 42nd Eur. Conf. Opt. Commun. (ECOC)*, 2016, pp. 1–3.

[102] F. Aubry *et al.*, "SCMon: Leveraging segment routing to improve network monitoring," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (IEEE INFOCOM)*, 2016, pp. 1–9.

[103] X. Li and K. L. Yeung, "Bandwidth-efficient network monitoring algorithms based on segment routing," *Comput. Netw.*, vol. 147, pp. 236–245, Dec. 2018.

[104] X. Li and K. Yeung, "ILP formulation for monitoring-cycle construction using segment routing," in *Proc. IEEE 43rd Conf. Local Comput. Netw. (LCN)*, 2018, pp. 485–492.

[105] M. Polverini, A. Cianfrani, M. Listanti, and A. Baiocchi, "Routing perturbation for traffic matrix evaluation in a segment routing network," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 4, pp. 1645–1660, Dec. 2018.

[106] M. P. A. Cianfrani and T. Nalawade, "A heuristic approach to assess the traffic matrix of an ISP exploiting segment routing flexibility," in *Proc. 30th Int. Teletraffic Congr. (ITC)*, vol. 1, 2018, pp. 194–199.

[107] A. C. M. Polverini and M. Listanti, "Interface counters in segment routing v6: A powerful instrument for traffic matrix assessment," in *Proc. IEEE 9th Int. Conf. Netw. Future (NOF)*, 2018, pp. 76–82.

[108] F. M. Xhonneux and O. Bonaventure, "Leveraging eBPF for programmable network functions with IPv6 segment routing," in *Proc. ACM 14th Int. Conf. Emerg. Netw. Exp. Technol.*, 2018, pp. 67–72.

[109] M. Lee and J. Sheu, "An efficient routing algorithm based on segment routing in software-defined networking," *Comput. Netw.*, vol. 103, pp. 44–55, Jul. 2016.

[110] A. Bahnasse and F. L. Louhab, "Novel SDN architecture for smart MPLS traffic engineering-DiffServ aware management," *Future Gener. Comput. Syst.*, vol. 87, pp. 115–126, Oct. 2018.

[111] J. Sheu and Y. Chen, "A scalable and bandwidth-efficient multicast algorithm based on segment routing in software-defined networking," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2017, pp. 1–6.

[112] O. G. R. Carpa and L. Lefevre, "Segment routing based traffic engineering for energy efficient backbone networks," in *Proc. IEEE Int. Conf. Adv. Netw. Telecommun. Syst. (ANTS)*, 2014, pp. 1–6.

[113] K. Ghuman and A. Nayak, "Per-packet based energy aware segment routing approach for data center networks with SDN," in *Proc. IEEE 24th Int. Conf. Telecommun. (ICT)*, 2017, pp. 1–6.

[114] L. Davoli, L. Veltri, P. L. Ventre, G. Siracusano, and S. Salsano, "Traffic engineering with segment routing: SDN-based architectural design and open source implementation," in *Proc. IEEE 4th Eur. Workshop Softw. Defined Netw. (EWSDN)*, 2015, pp. 111–112.

[115] G. Trimponias *et al.*, "On traffic engineering with segment routing in SDN based WANs," 2017. [Online]. Available: arXiv:1703.05907.

[116] R. Bhatia, F. Hao, M. S. Kodialam, and T. V. Lakshman, "Optimized network traffic engineering using segment routing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2015, pp. 657–665.

[117] H. Roomi and S. Khorsandi, "Semi-oblivious segment routing with bounded traffic fluctuations," in *Proc. IEEE Iran. Conf. Electr. Eng. (ICEE)*, 2018, pp. 1670–1675.

[118] R. Hartert *et al.*, "A declarative and expressive approach to control forwarding paths in carrier-grade networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 15–28, 2015.

[119] H. R. S. Gay and S. Vissicchio, "Expect the unexpected: Sub-second optimization for segment routing," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, 2017, pp. 1–9.

[120] R. Hartert, P. Schaus, S. Vissicchio, and O. Bonaventure, "Solving Segment Routing problems with hybrid constraint programming techniques," in *Proc. Int. Conf. Principles Practice Constraint Program.*, 2015, pp. 592–608.

[121] M. L. A. Cianfrani and M. Polverini, "Incremental deployment of segment routing into an ISP network: A traffic engineering perspective," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3146–3160, Oct. 2017.

[122] A. B. E. Moreno and F. Cugini, "Traffic engineering in segment routing networks," *Comput. Netw.*, vol. 114, pp. 23–31, Feb. 2017.

[123] M. R. V. Pereira and P. Sousa, "Optimizing segment routing using evolutionary computation," *Procedia Comput. Sci.*, vol. 110, pp. 312–319, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050917312784

[124] A. A. Barakabitze, L. Sun, I.-H. Mkwawa, and E. C. Ifeachor, "A novel QoE-Centric SDN-based multipath routing approach for multimedia services over 5G networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–7.

[125] J. Pang, G. Xu, and X. Fu, "SDN-based data center networking with collaboration of multipath TCP and segment routing," *IEEE Access*, vol. 5, pp. 9764–9773, 2017.

[126] O. Dugeon, R. Guedrez, S. Lahoud, and G. Texier, "Demonstration of segment routing with SDN based label stack optimization," in *Proc. IEEE 20th Conf. Innov. Clouds Internet Netw. (ICIN)*, 2017, pp. 143–145.

[127] X. Hou, M. Wu, and M. Zhao, "An optimization routing algorithm based on segment routing in software-defined networks," *Sensors*, vol. 19, no. 1, p. 49, 2019.

[128] T. Settawatcharawanit, V. Suppakitpaisarn, S. Yamada, and Y. Ji, "Segment routed traffic engineering with bounded stretch in software-defined networks," in *Proc. IEEE 43rd Conf. Local Comput. Netw. (LCN)*, 2018, pp. 477–480.

[129] G. Trimponias, Y. Xiao, X. Wu, H. Xu, and Y. Geng, "Node-constrained traffic engineering: Theory and applications," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1344–1358, Aug. 2019.

[130] P. Zhang, Y. Gang, X. Huang, S. Zeng, and K. Xie, "Bandwidth allocation with utility maximization in the hybrid segment routing network," *IEEE Access*, vol. 7, pp. 85253–85261, 2019.

[131] K.-T. Foerster, M. Parham, M. Chiesa, and S. Schmid, "TI-MFA: Keep calm and reroute segments fast," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2018, pp. 415–420.

[132] F. Aubry, D. Lebrun, Y. Deville, and O. Bonaventure, "Traffic duplication through segmentable disjoint paths," in *Proc. IEEE IFIP Netw. Conf. (IFIP Netw.)*, 2015, pp. 1–9.

[133] F. Aubry, S. Vissicchio, O. Bonaventure, and Y. Deville, "Robustly disjoint paths with segment routing," in *Proc. ACM 14th Int. Conf. Emerg. Netw. Exp. Technol.*, 2018, pp. 204–216.

[134] F. Hao, M. S. Kodialam, and T. V. Lakshman, "Optimizing restoration with segment routing," in *Proc. IEEE 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, 2016, pp. 1–9.

[135] A. Giorgetti, A. Sgambelluri, F. Paolucci, F. Cugini, and P. Castoldi, "Segment routing for effective recovery and multi-domain traffic engineering," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 9, no. 2, pp. A223–A232, Feb. 2017.

[136] A. Giorgetti, A. Sgambelluri, F. Paolucci, F. Cugini, and P. Castoldi, "Demonstration of dynamic restoration in segment routing multi-layer SDN networks," in *Proc. Opt. Fiber Commun. Conf.*, 2016, p. 4.

[137] A. Giorgetti, A. Sgambelluri, F. Paolucci, and P. Castoldi, "Reliable segment routing," in *Proc. IEEE 7th Int. Workshop Rel. Netw. Design Model. (RNDM)*, 2015, pp. 181–185.

[138] M. Xhonneux and O. Bonaventure, "Flexible failure detection and fast reroute using eBPF and SRv6," in *Proc. 14th Int. Conf. Netw. Service Manag. (CNSM)*, 2018, pp. 408–413.

[139] K. T. Foerster *et al.* (2018). *Local Fast Segment Rerouting on Hypercubes*. [Online]. Available: http://eprints.cs.univie.ac.at/5837/11/2018-hypercubes.pdf

[140] A. Sgambelluri, F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi, "First demonstration of SDN-based segment routing in multi-layer networks," in *Proc. IEEE Opt. Fiber Commun. Conf. Exhibit. (OFC)*, 2015, pp. 1–3.

[141] A. Sgambelluri, F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi, "SDN and PCE implementations for segment routing," in *Proc. IEEE 20th Eur. Conf. Netw. Opt. Commun. (NOC)*, 2015, pp. 1–4.

[142] F. Paolucci, "Network service chaining using segment routing in multi-layer networks," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 10, no. 6, pp. 582–592, Jun. 2018.

[143] F. Paolucci *et al.*, "Service chaining in multi-layer networks using segment routing and extended BGP FlowSpec," in *Proc. Opt. Fiber Commun. Conf.*, 2017, p. 3.

[144] F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi, "Segment routing in multi-layer networks," in *Proc. IEEE 19th Int. Conf. Transp. Opt. Netw. (ICTON)*, 2017, pp. 1–4.

[145] *Spring Open Project*. Accessed: Dec. 3, 2020. [Online]. Available: https://wiki.onosproject.org/display/ONOS10/Segment+Routing

[146] A. Fressancourt and M. Gagnaire, "A SDN-based network architecture for cloud resiliency," in *Proc. 12th Annu. IEEE Consum. Commun. Netw. Conf. (CCNC)*, 2015, pp. 479–484.

[147] Z. Li, L. Huang, H. Xu, and G. Zhao, "Segment routing in hybrid software-defined networking," in *Proc. IEEE 9th Int. Conf. Commun. Softw. Netw. (ICCSN)*, 2017, pp. 160–165.

[148] D. Lebrun, M. Jadin, F. Clad, C. Filsfils, and O. Bonaventure, "Software resolved networks: Rethinking enterprise networks with IPv6 segment routing," in *Proc. ACM Symp. SDN Res.*, 2018, p. 6.

[149] M. J. F. Duchene and O. Bonaventure, "Exploring various use cases for IPv6 segment routing," in *Proc. ACM SIGCOMM Conf. Posters Demos*, 2018, pp. 129–131.

[150] N. Kukreja *et al.*, "Demonstration of SDN-based orchestration for multi-domain segment routing networks," in *Proc. IEEE 18th Int. Conf. Transp. Opt. Netw. (ICTON)*, 2016, pp. 1–4.

[151] A. Sgambelluri, A. Giorgetti, F. Paolucci, F. Cugini, and P. Castoldi, "Experimental demonstration of multi-domain segment routing," in *Proc. IEEE Eur. Conf. Opt. Commun. (ECOC)*, 2015, pp. 1–3.

[152] D. Cai, A. Wielosz, and S. Wei, "Evolve carrier Ethernet architecture with SDN and segment routing," in *Proc. IEEE 15th Int. Symp. World Wireless Mobile Multimedia Netw. (WoWMoM)*, 2014, pp. 1–6.

[153] O. L. Barakat, P. L. Ventre, S. Salsano, and X. Fu, "Busoni: Policy composition and northbound interface for IPv6 segment routing networks," in *Proc. IEEE 27th Int. Conf. Netw. Protocols (ICNP)*, 2019, pp. 1–4.

[154] V. Eramo, F. G. Lavacca, T. Catena, M. Polverini, and A. Cianfrani, "Effectiveness of segment routing technology in reducing the bandwidth and cloud resources provisioning times in network function virtualization architectures," *Future Internet*, vol. 11, no. 3, p. 71, 2019.

[155] A. Sgambelluri, A. Giorgetti, F. Paolucci, F. Cugini, and P. Castoldi, "Experimental demonstration of segment routing," *J. Lightw. Technol.*, vol. 34, no. 1, pp. 205–212, Jan. 1, 2016.

[156] F. Lazzeri, G. Bruno, J. Nijhof, A. Giorgetti, and P. Castoldi, "Efficient label encoding in segment-routing enabled optical networks," in *Proc. IEEE Int. Conf. Opt. Netw. Design Model. (ONDM)*, 2015, pp. 34–38.

[157] A. Giorgetti, P. Castoldi, F. Cugini, J. Nijhof, F. Lazzeri, and G. Bruno, "Path encoding in segment routing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2015, pp. 1–6.

[158] R. Guedrez, O. Dugeon, S. Lahoud, and G. Texier, "Label encoding algorithm for MPLS segment routing," in *Proc. IEEE 15th Int. Symp. Netw. Comput. Appl. (NCA)*, 2016, pp. 113–117.

[159] M. L. A. Cianfrani and M. Polverini, "Translating traffic engineering outcome into segment routing paths: The encoding problem," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2016, pp. 245–250.

[160] L. Huang, Q. Shen, W. Shao, and X. Cui, "Optimizing segment routing with the maximum SLD constraint using Openflow," *IEEE Access*, vol. 6, pp. 30874–30891, 2018.

[161] R. Guedrez, O. Dugeon, S. Lahoud, and G. Texier, "A new method for encoding MPLS segment routing TE paths," in *Proc. IEEE 8th Int. Conf. Netw. Future (NOF)*, 2017, pp. 58–65.

[162] D. Lebrun, "Leveraging IPv6 segment routing for service function chaining," in *Proc. CoNEXT Stud. Workshop*, 2015, pp. 1–15.

[163] A. Abdelsalam, F. Clad, C. Filsfils, S. Salsano, G. Siracusano, and L. Veltri, "Implementation of virtual network function chaining through segment routing in a linux-based NFV infrastructure," in *Proc. IEEE Conf. Netw. Softw. (NetSoft)*, Jul. 2017, pp. 1–5.

[164] D. L. F. Duchêne and O. Bonaventure, "SRv6Pipes: Enabling in-network bytestream functions," in *Proc. IFIP Netw.*, 2018, pp. 37–45.

[165] A. Abdelsalam, S. Salsano, F. Clad, P. Camarillo, and C. Filsfils "SERA: Segment routing aware firewall for service function chaining scenarios," in *Proc. IEEE IFIP Netw.*, May 2018, pp. 46–54.

[166] Y. Desmouceaux, P. Pfister, J. Tollet, M. Townsley, and T. H. Clausen, "SRLB: The power of choices in load balancing with segment routing," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2017, pp. 2011–2016.

[167] Y. Desmouceaux, M. Enguehard, V. Nguyen, P. Pfister, W. Shao, and E. Vyncke, "A content-aware Data-plane for efficient and scalable video delivery," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manag. (IM)*, 2019, pp. 10–18.

[168] A. Abdelsalam *et al.*, "Performance of IPv6 segment routing in linux kernel," in *Proc. 1st Workshop Segment Routing Service Funct. Chaining (SR+SFC) CNSM*, Rome, Italy, 2018, pp. 414–419.

[169] HCL SRv6 Solution Team and C. Tato. *Segment Routing Over IPv6 Acceleration Using Intel FPGA Programmable Acceleration Card N3000*. Accessed: Dec. 3, 2020. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01295-hcl-segment-routing-over-ipv6-acceleration-using-intel-fpga-programmable-acceleration-card-n3000.pdf

[170] C. Lee, K. Ebisawa, H. Kuwata, M. Kohno, and S. Matsushima, "Performance evaluation of GTP-U and SRv6 stateless translation," in *Proc. 2nd Workshop Segment Routing Service Function Chaining (SR+SFC) CNSM*, 2019, pp. 1–6.

[171] A. Abdelsalam *et al.*, "SRPerf: A performance evaluation framework for IPv6 segment routing," 2020. [Online]. Available: arXiv:2001.06182.

[172] V. Kotronis, X. A. Dimitropoulos, R. Klöti, B. Ager, P. Georgopoulos, and S. Schmid, "Control exchange points: Providing QoS-enabled end-to-end services via SDN-based inter-domain routing orchestration," in *Proc. 3rd Open Netw. Summit (ONS)*, Santa Clara, CA, USA, Mar. 2014.

[173] P. B. Godfrey, S. Shenker, and I. Stoica, "Pathlet routing," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 111–122, 2009.

[174] S. Bidkar *et al.*, "Scalable segment routing: A new paradigm for efficient service provider networking using carrier Ethernet advances," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 7, no. 5, pp. 445–460, May 2015.

[175] S. Bidkar, A. Gumaste, and A. K. Somani, "A scalable framework for segment routing in service provider networks: The omnipresent Ethernet approach," in *Proc. IEEE 15th Int. Conf. High Perform. Switch. Routing (HPSR)*, 2014, pp. 76–83.

[176] S. Bidkar *et al.*, "Field trial of a Software Defined Network (SDN) using carrier Ethernet and segment routing in a tier-1 provider," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2014, pp. 2166–2172.

[177] T. Schuller, N. Aschenbruck, M. Chimani, and M. Horneffer, "On the practical irrelevance of metrics on segment routing traffic engineering optimization," in *Proc. IEEE 43rd Conf. Local Comput. Netw. (LCN)*, 2018, pp. 640–647.

[178] T.-H. Chi, C.-H. Lin, J.-J. Kuo, and W.-T. Chen, "Live video multicast for dynamic users via segment routing in 5G networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2018, pp. 1–7.

[179] J. Cao, X. Wang, M. Huang, and X. Zhou, "A mobility-supported routing mechanism in industrial IoT networks," *IEEE Access*, vol. 7, pp. 25603–25615, 2019.

[180] A. Mayer, E. Altomare, S. Salsano, F. L. Presti, and C. Filsfils, "The network as a computer with IPv6 segment routing: A novel distributed processing model for the Internet of Things," in *Proc. 1st Int. Workshop Next Gener. Oper. Syst. Cyber Phys. Syst. (NGOSCPS) CPS-IoT Week*, Apr. 2019, pp. 1–4.

[181] *SDN-TE-SR*. Accessed: Dec. 3, 2020. [Online]. Available: https://github.com/netgroup/SDN-TE-SR

[182] V. Pereira, M. Rocha, and P. Sousa, "Segment routing single link failure congestion optimization," in *Proc. ICETE*, vol. 1, 2018, pp. 242–249.

[183] D. Lebrun, "A linux kernel implementation of segment routing with IPv6," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2016, pp. 1019–1020.

[184] *How to Work With Fast-Failover OpenFlow Groups*. Accessed: Dec. 3, 2020. [Online]. Available: https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/7995427/How+to+Work+with+Fast Failover+OpenFlow+Groups

[185] *A Thorough Introduction to eBPF*. Accessed: Dec. 3, 2020. [Online]. Available: https://lwn.net/Articles/740157/

[186] D. Katz and D. Ward, "Bidirectional forwarding detection (BFD)," IETF, RFC 5880, Jun. 2010. [Online]. Available: https://tools.ietf.org/html/rfc5880/

[187] *RYU Project*. Accessed: Dec. 3, 2020. [Online]. Available: https://github.com/faucetsdn/ryu

[188] Open Networking Foundation. *ONOS Project*. Accessed: Dec. 3, 2020. [Online]. Available: https://onosproject.org

[189] Trellis. *Open Source SDN L2/L3 Spine Leaf Switching Fabric for Networking*. Accessed: Dec. 3, 2020. [Online]. Available: https://www.opennetworking.org/trellis

[190] *Open Compute Project*. Accessed: Dec. 3, 2020. [Online]. Available: http://www.opencompute.org

[191] Broadcom. *OpenFlow Datapath Abstraction*. Accessed: Dec. 3, 2020. [Online]. Available: https://github.com/Broadcom-Switch/of-dpa

[192] *Broadcom*. Accessed: Dec. 3, 2020. [Online]. Available: https://www.broadcom.com

[193] D. Lebrun and O. Bonaventure, "Implementing IPv6 segment routing in the Linux kernel," in *Proc. ACM Appl. Netw. Res. Workshop*, 2017, pp. 35–41.

[194] S. Bradner, "Benchmarking terminology for network interconnection devices," IETF, RFC 1242, Jul. 1991. [Online]. Available: https://tools.ietf.org/html/rfc1242

[195] *Linux Foundation Wiki—Iproute2*. Accessed: Dec. 3, 2020. [Online]. Available: https://wiki.linuxfoundation.org/networking/iproute2

[196] *SRv6—Linux Kernel Implementation—Basic Configuration*. Accessed: Dec. 3, 2020. [Online]. Available: https://segment-routing.org/index.php/Implementation/Configuration

[197] D. Lebrun. (2017). *Reaping the Benefits of IPv6 Segment Routing*. [Online]. Available: https://inl.info.ucl.ac.be/system/files/phdthesis-lebrun.pdf

[198] *SRv6—Linux Kernel Implementation—Advanced Configuration*. Accessed: Dec. 3, 2020. [Online]. Available: https://segment-routing.org/index.php/Implementation/AdvancedConf

[199] *Linux Kernel Newbies—Linux 4.16*. Accessed: Dec. 3, 2020. [Online]. Available: https://kernelnewbies.org/Linux_4.16

[200] *Linux Netfilter Hacking*. Accessed: Dec. 3, 2020. [Online]. Available: https://www.netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO-3.html

[201] *Arch Wiki—Iptables*. Accessed: Dec. 3, 2020. [Online]. Available: https://wiki.archlinux.org/index.php/iptables

[202] *Linux Kernel Newbies—Linux 4.18*. Accessed: Dec. 3, 2020. [Online]. Available: https://kernelnewbies.org/Linux_4.18

[203] *Programming Network Actions With BPF*. Accessed: Dec. 3, 2020. [Online]. Available: https://segment-routing.org/index.php/Implementation/BPF

[204] *SRv6-BPF*. Accessed: Dec. 3, 2020. [Online]. Available: https://github.com/Zashas/Thesis-SRv6-BPF

[205] *SRv6-BFD*. Accessed: Dec. 3, 2020. [Online]. Available: https://github.com/Zashas/SRv6-BFD

[206] *What Is VPP?* Accessed: Dec. 3, 2020. [Online]. Available: https://wiki.fd.io/view/VPP

[207] *SRv6-Net-Prog*. Accessed: Dec. 3, 2020. [Online]. Available: https://netgroup.github.io/SRv6-net-prog/

[208] *FRRouting*. Accessed: Dec. 3, 2020. [Online]. Available: https://frrouting.org

[209] *Quagga Project*. Accessed: Dec. 3, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Quagga_(software)

[210] *OSPFv2 Segment Routing*. Accessed: Dec. 3, 2020. [Online]. Available: https://github.com/FRRouting/frr/blob/master/doc/developer/ospf-sr.rst

[211] *P4Language Consortium*. Accessed: Dec. 3, 2020. [Online]. Available: https://p4.org

[212] *P4Runtime*. Accessed: Dec. 3, 2020. [Online]. Available: https://p4.org/p4-runtime/

[213] *Stratum Project*. Accessed: Dec. 3, 2020. [Online]. Available: https://stratumproject.org

[214] *OpenCORD Project*. Accessed: Dec. 3, 2020. [Online]. Available: https://opencord.org

[215] *SEBA Project*. Accessed: Dec. 3, 2020. [Online]. Available: https://guide.opencord.org/profiles/seba/

[216] *Trellis Documentation*. Accessed: Dec. 3, 2020. [Online]. Available: https://docs.trellisfabric.org

[217] *Trellis Tutorial*. Accessed: Dec. 3, 2020. [Online]. Available: http://bit.ly/trellis-p4-slides

[218] S. Salsano, P. L. Ventre, L. Prete, G. Siracusano, M. Gerola, and E. Salvadori, "OSHI—Open source hybrid IP/SDN networking (and its emulation on Mininet and on distributed SDN testbeds)," in *Proc. IEEE 3rd Eur. Workshop Softw. Defined Netw. (EWSDN)*, 2014, pp. 13–18.

[219] S. Salsano *et al.*, "Hybrid IP/SDN networking: Open implementation and experiment management tools," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 1, pp. 138–153, Mar. 2016.

[220] *Open vSwitch*. Accessed: Dec. 3, 2020. [Online]. Available: http://openvswitch.org

[221] S. Salsano *et al.*, "Mantoo—A set of management tools for controlling SDN experiments," in *Proc. IEEE 4th Eur. Workshop Softw. Defined Netw. (EWSDN)*, 2015, pp. 123–124.

[222] *PMSR Tutorial*. Accessed: Dec. 3, 2020. [Online]. Available: https://github.com/netgroup/SDN-TE-SR-tools

[223] *OSHI Homepage*. Accessed: Dec. 3, 2020. [Online]. Available: http://netgroup.uniroma2.it/OSHI

[224] B. Pfaff and B. Davie, "The Open vSwitch database management protocol," IETF, RFC 7047, Dec. 2013. [Online]. Available: https://rfc-editor.org/rfc/rfc7047.txt

[225] *SRN*. Accessed: Dec. 3, 2020. [Online]. Available: https://github.com/segment-routing/srn

[226] *SRN Overview*. Accessed: Dec. 3, 2020. [Online]. Available: https://segment-routing.org/index.php/SRN/SRN

[227] *Using SRN*. Accessed: Dec. 3, 2020. [Online]. Available: https://segment-routing.org/index.php/SRN/UsingSRN

[228] *SRv6-SDN*. Accessed: Dec. 3, 2020. [Online]. Available: https://netgroup.github.io/srv6-sdn/

[229] *ROSE*. Accessed: Dec. 3, 2020. [Online]. Available: https://netgroup.github.io/rose/

[230] *SRv6Pipes*. Accessed: Dec. 3, 2020. [Online]. Available: https://github.com/segment-routing/SRv6Pipes

[231] *SRv6Pipes Walkthrough*. Accessed: Dec. 3, 2020. [Online]. Available: https://segment-routing.org/index.php/SRv6Pipes/SRv6Pipes

[232] *SRNK*. Accessed: Dec. 3, 2020. [Online]. Available: https://netgroup.github.io/srnk/

[233] *RobustlyDisjointPathsCode*. Accessed: Dec. 3, 2020. [Online]. Available: https://bitbucket.org/franaubry/robustlydisjointpathscode

[234] S. Matsushima, Z. Li, C. Filsfils, and Z. Ali, "SRv6 implementation and deployment status," Internet Eng. Task Force, Fremont, CA, USA, Internet-Draft draft-matsushima-spring-srv6-deployment-status, Apr. 2020. [Online]. Available: https://tools.ietf.org/html/draft-matsushima-spring-srv6-deployment-status

[235] *Barefoot Networks*. Accessed: Dec. 3, 2020. [Online]. Available: https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series.html

[236] *BMv2 Switch*. Accessed: Dec. 3, 2020. [Online]. Available: http://www.bmv2.org

[237] *Cavium*. Accessed: Dec. 3, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Cavium

[238] *SRv6 Interop Demo*. Accessed: Dec. 3, 2020. [Online]. Available: https://blogs.cisco.com/sp/segment-routing-ipv6-interoperability-demo-is-already-there

[239] EANTC. (Apr. 2018). *MPLS+SDN+NFVVORD@PARIS2018 Interoperability Showcase*. [Online]. Available: http://www.eantc.de/fileadmin/eantc/downloads/events/2017–2020/MPLS2018/EANTC-MPLSSDNNFV2018-WhitePaper-final.pdf

[240] *Wireshark*. Accessed: Dec. 3, 2020. [Online]. Available: https://www.wireshark.org

[241] *TCPDUMP*. Accessed: Dec. 3, 2020. [Online]. Available: http://www.tcpdump.org

[242] *Iptables*. Accessed: Dec. 3, 2020. [Online]. Available: https://www.netfilter.org

[243] *Nftables*. Accessed: Dec. 3, 2020. [Online]. Available: https://wiki.nftables.org/wiki-nftables/index.php/Main_Page

[244] *Snort*. Accessed: Dec. 3, 2020. [Online]. Available: https://www.snort.org

[245] *Open Programmable Acceleration Engine (OPAE)*. Accessed: Dec. 3, 2020. [Online]. Available: https://opae.github.io

[246] *Express Data Path*. Accessed: Dec. 3, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Express_Data_Path

[247] *OpenStack*. Accessed: Dec. 3, 2020. [Online]. Available: https://www.openstack.org

[248] *Kubernetes*. Accessed: Dec. 3, 2020. [Online]. Available: http://kubernetes.io

[249] P. Ventre *et al.*, "Segment routing: A comprehensive survey of research activities, standardization efforts and implementation results," 2019. [Online]. Available: arXiv:1904.03471.

Antonio Cianfrani (Member, IEEE) received the master's degree in telecommunications engineering and the Ph.D. degree in information and communication engineering from the University of Rome La Sapienza in 2004 and 2008, respectively, where he is currently an Associate Professor with the DIET Department. His fields of interest include routing algorithms, network protocols, performance evaluation of software routers, and green networks. His current research interests are focused on segment routing and traffic matrix assessment. He is the coauthor of more than 70 peer-reviewed papers and book chapters. He serves on the editorial boards of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING.

Ahmed Abdelsalam received the Ph.D. degree in computer science from Gran Sasso Science Institute in 2020. He is a Software Engineer with the IPv6 Segment Routing "SRv6" Architecture Team, Cisco. He contributed to many open source, including the Linux kernel, FD.IO VPP, Tcpdump, IPTables, and Snort. His main research interests focus on IPv6 segment routing (SRv6), network function virtualization, software-defined networks, service function chaining, and containers networking.

Pier Luigi Ventre received the Ph.D. degree in electronics engineering from the University of Rome "Tor Vergata" in 2018. He is a member of Technical Staff with the Open Networking Foundation (ONF), where he works on Trellis—the leading open-source leaf-spine fabric. Before joining ONF, he worked with CNIT as a Researcher and a Postdoctoral Researcher on several projects funded by the EU. From 2013 to 2015, he was one of the beneficiaries of the scholarship "Orio Carlini" granted by the Italian NREN GARR. His main interests focus on software-defined networking, network function virtualization, virtualization, and segment routing.

Clarence Filsfils received the Ph.D. degree in engineering science, the Masters of Management degree from Solvay Business School, and the Masters of Engineering degree in computer science from the University of Liege. He is a Cisco Systems Fellow and has a 25-year expertise leading innovation, productization, marketing, and deployment for Cisco Systems. He invented the Segment Routing Technology and is leading its productization, marketing, and deployment. Previously, he invented and led the Fast Routing Convergence Technology and was the Lead Designer for Cisco System's QoS deployments. He is a regular speaker at leading industry conferences. He holds over 195 patents and is a prolific writer, both in academic circles and in the standardization or books.

Stefano Salsano (Senior Member, IEEE) received the Ph.D. degree from the University of Rome "La Sapienza" in 1998. He is an Associate Professor with the University of Rome Tor Vergata. Since July 2018, he has been the Coordinator of the Bachelor's Degree "Ingegneria di Internet" and the Master's Degree "ICT and Internet Engineering". He has participated in 16 research projects funded by the EU, being a Project Coordinator for one of them and a Technical Coordinator for two. He has been a PI of several research and technology transfer contracts funded by industries. He is the coauthor of an IETF RFC and more than 170 peer-reviewed papers and book chapters. His current research interests include SDN, network virtualization, cybersecurity, and information-centric networking.

Pablo Camarillo is one of the engineers behind Segment Routing v6 at Cisco. He is the coauthor of various IETF drafts, holds several patents, and has developed SR implementation in FD.io VPP. Prior to joining Cisco, he was a Research Engineer with the IMDEA Networks Institute, where he prototyped a BGP route server in ExaBGP and researched the algorithmic of TI-LFA (SR Topology-Independent Loop-Free Alternates).

Marco Polverini received the master's degree in telecommunications engineering and the Ph.D. degree in information and communication engineering from the University of Rome La Sapienza in 2010 and 2014, respectively, where he is currently a Research Fellow with the Department of Information, Electronic and Telecommunications Engineering. His main research interests are routing protocols for energy saving in IP networks, network traffic monitoring, and measurement in next-generation routing technologies.

Francois Clad received the M.Sc. and Ph.D. degrees in computer science from the University of Strasbourg, France, in 2011 and 2014, respectively. He spent one year as a Postdoctoral Researcher with the Institute IMDEA Networks, Madrid, Spain, before joining Cisco in 2015. His research activities are focused on IP routing and in particular evolving the segment routing technology.