Experimental Demonstration of Segment Routing

A. Sgambelluri, F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi

***Abstract*—Segment Routing (SR) technology has been recently proposed to enforce effective routing strategies without relying on signaling protocols.**

So far, the SR technology has received limited attention within the scientific community.

In this paper, two SR implementations are presented and successfully demonstrated in two different network testbeds. The first implementation focuses on a Software Defined Networking (SDN) scenario where nodes consist of OpenFlow switches and the SR Controller is a specifically designed enhanced version of an OpenFlow Controller. The second implementation includes a novel PCE scenario where nodes consist of commercially available IP/MPLS routers and the SR Controller is a new extended version of a PCE solution.

Both implementations have been successfully applied to demonstrate dynamic traffic rerouting. In particular, by enforcing different segment list configurations at the ingress node, rerouting is effectively achieved with no packet loss and without requiring the use of signaling protocols. Effective scalability performance is achieved in both proposed implementations, under different segment list conditions.

I. Introduction

The Segment Routing (SR) technology has been recently introduced to enable effective Traffic Engineering (TE) solutions while simplifying control plane operations [1]–[3].

SR can be applied to Multiprotocol Label Switching (MPLS) networks with no change on the forwarding plane. SR relies on the source routing paradigm to enforce a packet flow through a path by applying, at the ingress node, a specifically designed stack of labels (i.e., the *segment list*) compatible with the MPLS data plane, that consists in an ordered list of segment identifiers (SIDs). During packet forwarding, only the top label in the segment list is processed. That is, the packet is forwarded along the shortest path toward the network element represented by the top label. For example, a SID can represents an IGP-Prefix, e.g. the loopback address of a node.

Unlike traditional MPLS networks, SR maintains per-flow state only at the ingress node, where the segment list is initialized. No signaling protocol is required to support packet forwarding in transit nodes, thus reducing the control plane load and simplifying the complex and time-consuming provisioning procedure, e.g., using Resource Reservation Protocol with TE extensions (RSVP-TE) [4]. This potential advantages can be particularly significant in multi-region/layer networks, where SR can eliminate the need to establish and maintain hierarchical instances of Label Switched Paths (LSPs) [5], [6]. Such Generalized MPLS (GMPLS) hierarchical approach has never reached adequate consideration for actual deployment, due to the introduced complexity of the control plane, overloaded with an excessive amount of signaling sessions. On the other hand, the SR technology, by avoiding signaling sessions at the packet layer, has the potential to enable lightweight control plane solutions also when multi-region/layer networks are considered. SR is gaining interest within the industrial community [7], [8]. However, the SR technology has received limited attention within the scientific community. In particular, to the best of our knowledge, only the following two scientific studies focus on SR. In [9] an SR solution derived from a modified Carrier Ethernet implementation, called omnipresent Ethernet, is proposed. The basic concept of this solution is the encoding of the labels, enabling binary routing through any interface of the network. The solution, to address possible scalability issues, also relies on a hierarchical architecture applied to a network that is partitioned into clusters. Also the work in [10] considers the SR application in Carrier Ethernet networks. In particular, the authors present a discussion on the benefits of the Carrier Ethernet technology enhanced to support SR and employing the Software Defined Networking (SDN) architecture. The work in [11] proposes a graph model that can be used to compute the segment list encoding a given path considering different constraints.

Differently with respect to the aforementioned studies, in this paper the SR technology is implemented and applied on two different multi-layer network scenarios. Two novel SR implementations are presented and discussed. In the first SR implementation, an OpenFlow-based SDN solution is proposed and validated using a data plane composed of OpenFlow switches and optical nodes. Specifically, the OpenFlow protocol [12]–[14] is here utilized for the first time to control the SR segment list configuration. In the second SR implementation, the Path Computation Element architecture is considered [15] where the PCE Protocol (PCEP) is extended and validated for the first time to support SR operation. In this case the data plane is composed of commercially available routers and optical nodes.

The two implementations rely on a specifically designed SR Controller architecture and on a common algorithm designed to provide the minimum depth segment list encoding a given path. Both implementations are successfully validated on a multi-layer network scenario, providing dynamic traffic adaptations without requiring multi-layer GMPLS operation [16], [17]. Both implementations are successfully validated, also assessing the scalability performance of the SR-based solutions.

II. Segment Routing operation

SR is typically associated with a centralized control plane implementation, where a single controller (i.e., the *SR controller*) is in charge of all the TE decisions in the network [10],

Manuscript received June 01, 2015; revised July 29, 2015.

A. Sgambelluri (email: a.sgambelluri@sssup.it), F. Paolucci, A. Giorgetti, and P. Castoldi are with Scuola Superiore Sant'Anna, Pisa, Italy.

F. Cugini, is with CNIT, Pisa, Italy.

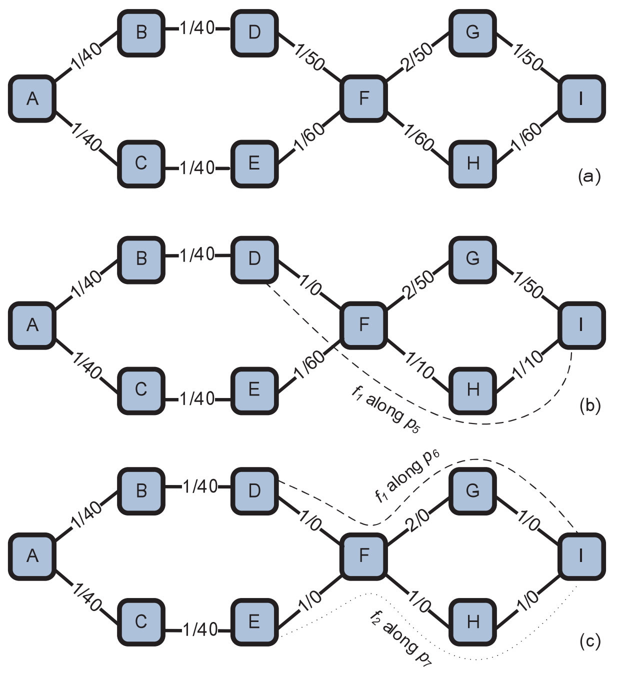


Fig. 1: Segment Routing network example and use-case.

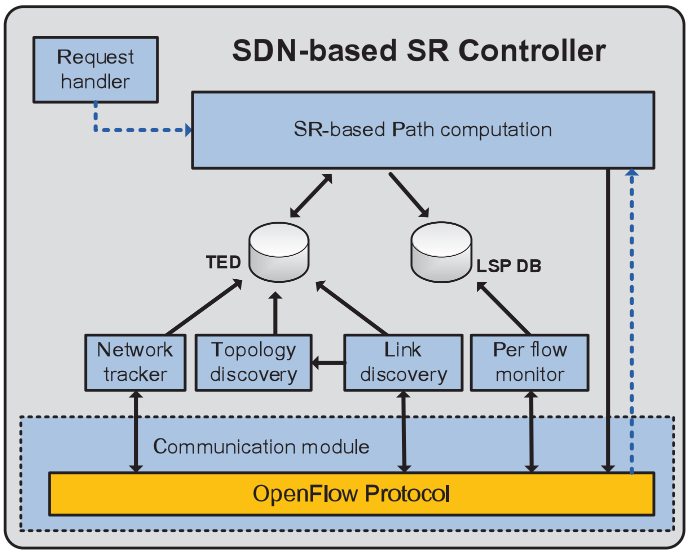


Fig. 2: SDN-based SR controller architecture.

[13], [18]. In this scenario, when a new traffic flow has to be established, a request is sent to the SR controller that performs the path computation and the encoding of the computed path using the proper segment list. The computed segment list is sent back to the node issuing the request and is applied to all packets belonging to the specific traffic flow. In particular, the segment list is determined assuming that all the nodes in the network will forward the packet using MPLS rules, i.e., looking at the top SID and forwarding the packet on the interface associated with that SID in the forwarding table.

Fig. 1(a) illustrates a simple network topology to clarify the main SR concepts, the two numbers reported on each link represent the metric cost and the available bandwidth unit, respectively. When a traffic flow has to be routed along the shortest path to its destination a segment list including only one label can be used (i.e., the SID of the destination node). As an example, in Fig. 1(a), if the target path for an incoming traffic flow is $\bar{p}_1 = \{D, F, H\}$ the segment list is $\overline{SL}^{\bar{p}_1} = \{H\}$. Conversely, if the target path is not the shortest path to the destination, a more complex segment list has to be used, e.g., if the desired path is $\bar{p}_2 = \{D, F, G, I, H\}$ a possible segment list is $\overline{SL}^{\bar{p}_2} = \{G, H\}$. Indeed, considering the illustrated cost metric the shortest path from D to G is the segment $\bar{s}_1 = \{D, F, G\}$ that is part of the target path $\bar{p}_1$; and the shortest path from G to H is the segment $\bar{s}_2 = \{G, I, H\}$ that stitched to $\bar{s}_1$ provides the target path $\bar{p}_2 = \{\bar{s}_1, \bar{s}_2\}$.

SR intrinsically supports load balancing among Equal Cost Multi Paths (ECMPs). As an example in Fig. 1(a) if a traffic flow from A to F is established using the segment list $\overline{SL} = \{F\}$ the traffic will be load balanced among the two paths $\bar{p}_3 = \{A, B, D, F\}$ and $\bar{p}_4 = \{A, C, E, F\}$. To distinguish among ECMPs a more detailed segment list is required, e.g., if the target path is $\bar{p}_3$ a possible segment list is $\overline{SL}^{\bar{p}_3} = \{B, F\}$.

Fig. 1(b) and Fig. 1(c) illustrate a possible SR use case. Specifically, in Fig. 1(b) the traffic flow f_1 requiring 50 units of bandwidth has been established along the path $\bar{p}_5 = \{D, F, H, I\}$. In this scenario, a new traffic flow f_2 requiring 60 units of bandwidth from node E to node I cannot be established due to lack of bandwidth. One solution could be the re-routing of f_1 to the path $\bar{p}_6 = \{D, F, G, I\}$, dashed in Fig. 1(c), so that f_2 can be established along the path $\bar{p}_7 = \{E, F, H, I\}$, dotted in Fig. 1(c). However, the re-routing procedure using RSVP-TE requires not negligible time and may imply loss of traffic if make-before-break [4] is not applicable due to lack of available bandwidth (i.e., the scenario in Fig. 1(c)). Conversely, by using SR, the SR controller can simply re-route f_1 from $\bar{p}_5$ to $\bar{p}_6$ sending to node D a new segment list (i.e., $\overline{SL}^{\bar{p}_6} = \{G, I\}$) to be enforced. This way, f_1 is re-routed in a negligible time without loss of traffic, and f_2 can be established along $\bar{p}_7$ using $\overline{SL}^{\bar{p}_7} = \{H, I\}$.

III. Proposed SR Controller Architecture

This section details the two different SR controller implementations: SDN-based and PCE-based.

The architecture of the implemented SDN-based SR controller is shown in Fig. 2, whereas the PCE-based is depicted in Fig. 3. The two architectures present some common modules: the *request handler*, the *SR-based path computation*, two databases (i.e., TED and LSP DB) and the *communication module*. The request handler provides the north bound interface to submit new requests. The SR-based path computation module is the core element of the system that processes the requests, selects the path, computes the segment list to be applied to the packets flow and updates the databases with proper information. The LSP-DB records the computed segment list and the route along the network for each relevant active traffic demand. The TED stores the network topology and network resource availability information. Specifically, the relevant traffic demands included in the LSP-DB are considered to account for the amount of utilized bandwidth in each traversed link. Such solution, derived from the stateful PCE architecture enables constrained based path computation [15]. The communication module enables the SR Controller connection with the data plane nodes. In particular in the SDN-based scheme the communication module exploits the OpenFlow protocol. Whereas in the PCE-based architecture, the communication module relies on the PCEP protocol.

In the SDN-based architecture more modules are required. In particular the *per flow monitor* has been designed to collect the statistics of installed flows; the *link discovery* module is used to discover the network links; the *topology discovery* module, relying on the information provided by the link discovery module, is used to detect the network topology (in a graph form) and to insert those information in the TED, and the *network tracker* module is used to keep track of the addresses of connected networks.

In Fig. 2 and Fig. 3 the information exchange flows among the architectural elements are represented with solid black lines. On the other hand, blue dashed lines represent the possible ways to submit new requests to the SR controller. With both architectures new requests can be received from the request handler or directly from the data plane. Specifically, new requests arriving from the data plane use an `OFP_PACKET_IN` OpenFlow message with the SDN-based controller, whereas `PCReq` PCEP message is used with the PCE-based controller.

A. Path and Segment List computation

The SR-based path computation module performs both the path computation and the encoding of the computed path with a specific segment list.

Regarding the path computation no specific algorithms are here proposed. Indeed the concept of SR can be applied independently on the used routing algorithm. In this case we apply a well known least congested path on a set of pre-computed and pre-validated candidate paths [19], [20]. Specifically, for each node pair (s, d) the set of candidate paths is indicated $P_{s,d}$ and includes all the feasible paths within one hop from the shortest path in terms of hops.

This work proposes an algorithm to compute a segment list encoding a given path using the minimum number of labels. Indeed, currently deployed MPLS equipment typically support a limited number of stacked labels, called *segment list depth*. The algorithms is compliant with the on-going standardization and provides the segment list of minimum depth when a unique path has to be used by avoiding load balancing among possible ECMPs.

In the algorithm description, the following notation is used: $\overline{SL}$, $\bar{p}$, and $\bar{s}_{i,j}$ are vectors of nodes in the general form $\bar{v} = \{v_0, v_1, \ldots, v_{n-1}\}$ including $|\bar{v}|$ elements. Specifically, $\overline{SL}$ is the output of the algorithm and represents the computed segment list; $\bar{p}_{s,d}$ is the target path from source node s to destination node d; $\bar{s}_{i,j}$ is the *target segment*, i.e., a portion of $\bar{p}$ including all nodes from p_i to p_j.

Fig. 4 describes the proposed segment list computation algorithm. The indexes i and j are managed to navigate the target path from source to destination. Specifically, the first

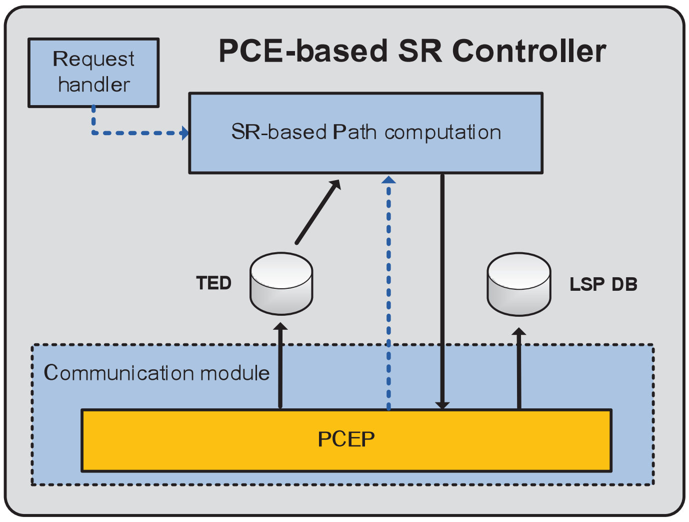


Fig. 3: PCE-based SR controller architecture.

evaluated target segment (i.e., $\bar{s}_{0,1}$) only includes the first two nodes of the target path, the second one (i.e., $\bar{s}_{0,2}$) includes the first three nodes and so forth. After proper initialization of indexes, if p_j is equal to the destination node d, the algorithm inserts d in the segment list and terminates. Otherwise, if the current target segment $s_{i,j}$ is a unique shortest path from p_i to p_j, an additional node is included from the target segment, i.e., index j is incremented, and the procedure is iterated. If not, the node p_{j-1} is included in the segment list and the indexes are configured so that the next target segment $s_{i,j}$ starts at p_{j-1} which is the last label inserted in the segment list.

As an example, with reference to Fig. 1, if the algorithm is used to encode the target path $\bar{p}_8 = \{A, B, D, F, H, I\}$, the obtained segment lists is $\overline{SL}^{\bar{p}_8} = \{B, I\}$. The provided segment list is of minimum depth although, in the considered example there is also an other segment list of minimum depth, i.e., $\overline{SL}^{\bar{p}_8} = \{D, I\}$.

IV. SR Controller implementation

A. SDN-based SR implementation

According to the architecture presented in Sec. III, the SDN-based SR Controller has been developed by enhancing the OpenFlow RYU controller [21]. To support the MPLS technology the controller relies on the OpenFlow 1.3 [22]. When a new request is received by the request handler, the SR-based path computation module collects the required information from the databases. Specifically, the request contains a type field, the source network, the destination network and an average bandwidth value. Initially the controller discovers the location of source and destination networks, then computes a path to be used in the network, keeping into account the requested bandwidth and the information stored in the TED. The final segment list is obtained by running the algorithms described in Sec. III on the selected path. Then exploiting the OpenFlow communication module the SR Controller sends a specific `OFP_FLOW_MOD` message to properly configure the ingress node. In particular a new flow-entry is installed: the match is made considering the input port

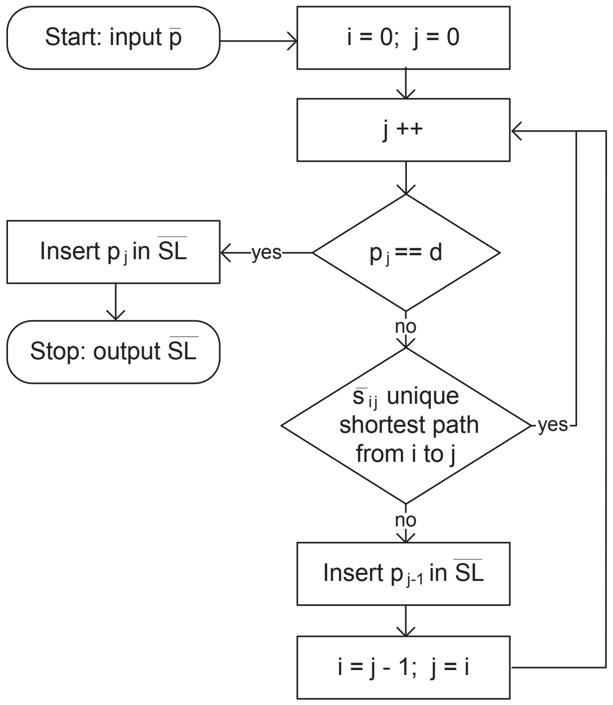
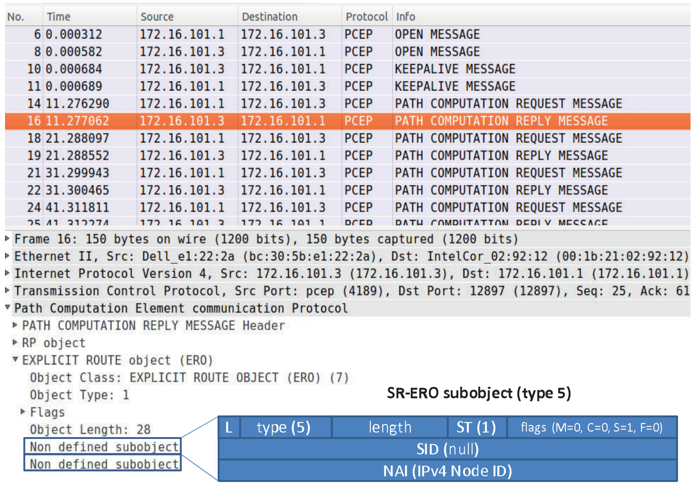

Start: input $\bar{p}$
i = 0; j = 0
j ++
p_j == d
yes
Insert p_j in $\overline{SL}$
Stop: output $\overline{SL}$
no
s_{ij} unique shortest path from i to j
yes
no
Insert p_{j-1} in $\overline{SL}$
i = j - 1; j = i

Fig. 4: Segment list computation algorithm.



```
No.  Time       Source        Destination   Protocol Info
  6 0.000312    172.16.101.1  172.16.101.3  PCEP     OPEN MESSAGE
  8 0.000582    172.16.101.3  172.16.101.1  PCEP     OPEN MESSAGE
 10 0.000684    172.16.101.3  172.16.101.1  PCEP     KEEPALIVE MESSAGE
 11 0.000689    172.16.101.1  172.16.101.3  PCEP     KEEPALIVE MESSAGE
 14 11.276290   172.16.101.1  172.16.101.3  PCEP     PATH COMPUTATION REQUEST MESSAGE
 16 11.277062   172.16.101.3  172.16.101.1  PCEP     PATH COMPUTATION REPLY MESSAGE
 18 21.288097   172.16.101.1  172.16.101.3  PCEP     PATH COMPUTATION REQUEST MESSAGE
 19 21.288552   172.16.101.3  172.16.101.1  PCEP     PATH COMPUTATION REPLY MESSAGE
 21 31.299943   172.16.101.1  172.16.101.3  PCEP     PATH COMPUTATION REQUEST MESSAGE
 22 31.300465   172.16.101.3  172.16.101.1  PCEP     PATH COMPUTATION REPLY MESSAGE
 24 41.311811   172.16.101.1  172.16.101.3  PCEP     PATH COMPUTATION REQUEST MESSAGE
 25 41.312274   172.16.101.3  172.16.101.1  PCEP     PATH COMPUTATION REPLY MESSAGE
▸ Frame 16: 150 bytes on wire (1200 bits), 150 bytes captured (1200 bits)
▸ Ethernet II, Src: Dell_e1:22:2a (bc:30:5b:e1:22:2a), Dst: IntelCor_02:92:12 (00:1b:21:02:92:12)
▸ Internet Protocol Version 4, Src: 172.16.101.3 (172.16.101.3), Dst: 172.16.101.1 (172.16.101.1)
▸ Transmission Control Protocol, Src Port: pcep (4189), Dst Port: 12897 (12897), Seq: 25, Ack: 61
▾ Path Computation Element communication Protocol
  ▸ PATH COMPUTATION REPLY MESSAGE Header
  ▸ RP object
  ▾ EXPLICIT ROUTE object (ERO)
     Object Class: EXPLICIT ROUTE OBJECT (ERO) (7)
     Object Type: 1
   ▸ Flags
     Object Length: 28
     Non defined subobject
     Non defined subobject
```

Fig. 5: Wireshark capture of PCEP session between SR Controller and the PCC edge source node.

and the destination IP address; the action is composed by a list of operation pushing/setting all the required MPLS labels. To reduce the ingress node setup time, the *instruction* feature of the `OFP_FLOW_MOD` message is exploited. In this way all the required actions are sent to the switch using a single message.

B. PCE-based SR implementation

In this section, the SR Controller architecture described in Sec. III and represented in Fig. 3 is implemented in the context of the PCE architecture.

PCEP sessions are established between the LSP source node acting as path computation client (PCC) and the SR Controller acting as PCE. The TED and the LSP-DB are updated by means of dedicated PCEP messages upon LSP establishment and release. The PCEP protocol is here extended to support SR, as proposed in [23]. In particular, the OPEN object includes the SR-PCE-CAPABILITY TLV, specifying the capability of handling SR-enabled LSPs requests by the PCC and the capability to perform segment list encoding at the PCE. Moreover, the Explicit Routing Object (ERO) carried out in the `PCRep` message encloses the computed segment list expressed as a list of Nodal Adjacency Identifiers (NAIs), that in this work correspond to the IP addresses of the considered nodes. The PCC is then in charge of matching the provided NAIs onto the corresponding SIDs by querying its local TED.

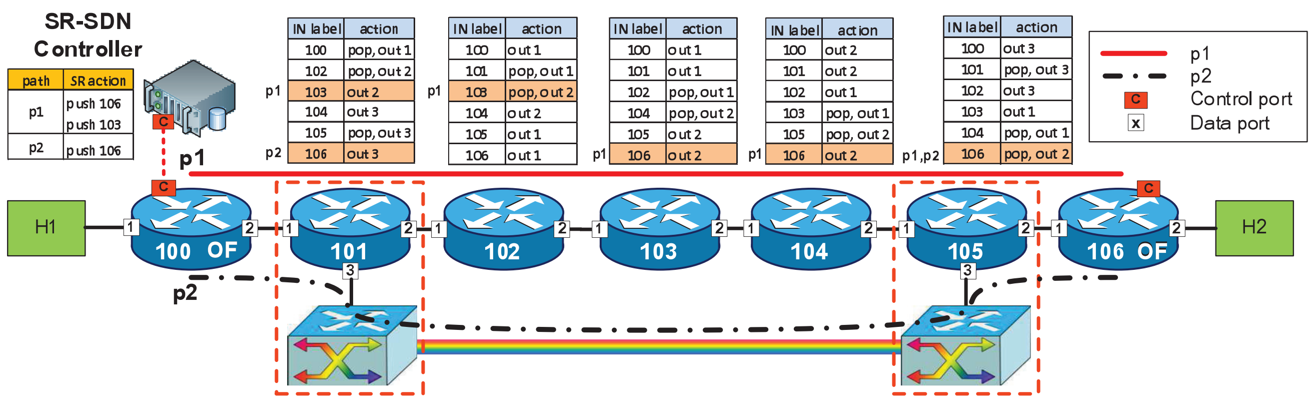
Fig. 5 shows a Wireshark capture at the SR Controller detailing the PCEP session between the PCE (IP address 172.16.101.3) and the PCC (IP address 172.16.101.1). The details of `PCRep` message are highlighted. In particular, the novel SR-ERO sub-object specifies the SID type (ST set to 1, IPv4 Node ID) and the flag S is set (i.e., the SID value is null). The loose flag L is not set.

V. Data plane implementation

A. Reference scenario

Fig. 6 shows the considered multi-layer packet over optical network reference scenario. The network includes 7 nodes equipped with packet switching capable (PSC) interfaces (i.e., Gigabit Ethernet optical interfaces) supporting MPLS forwarding. In particular, the PSC interfaces support pop and push operations on MPLS labels. Such operations are utilized during SR-based forwarding. The routing tables within the PSC nodes enable the selection of the next hop along the shortest path. This way, each node is able to autonomously determine the forwarding table used by the SR technology, i.e. based on SID. The edges of the PSC domain (i.e., node 100 and 106 in Fig. 6) support SR configurations provided by an SR controller. In the considered scenario, the network is multi-layer as it includes two optical nodes, i.e., two reconfigurable optical add/drop multiplexers (ROADMs) equipped with 10Gb/s Optical Transport Network (OTN) muxponders. These optical nodes are transparent with respect to SR forwarding. Specifically, a pre-established lightpath between node 101 and node 105 is utilized to enable the optical bypass from node 101 to node 105.

Fig. 6 also shows the forwarding operations configured at each node according to the possible top SIDs. Penultimate hop popping is considered, therefore, the forwarding table at node 101 specifies the pop operation for all packets having top SID value indicating an adjacent node (i.e., nodes 100, 102 and 105). Conversely, a simple forwarding (i.e., without swapping the top label) is specified for packets indicating on the top SID a non-adjacent node. In all cases, the outgoing port is identified along the shortest path. The push operations configured by the SR Controller at the ingress node (i.e., node 100) enforce the computed segment list thus enabling the definition of the target path. In the considered network scenario, a single push operation with top SID 106 identifies the shortest path, i.e, using the optical bypass. Conversely, a stack of two labels with 103 at the top and 106 at the bottom enables the selection of the route through the sequence of PSC nodes 102 – 104,



path	SR action
p1	push 106 push 103
p2	push 106

IN label	action
100	pop, out 1
102	pop, out 2
103	out 2
104	out 3
105	pop, out 3
106	out 3

IN label	action
100	out 1
101	pop, out 1
103	pop, out 2
104	out 2
105	out 1
106	out 1

IN label	action
100	out 1
101	out 1
102	pop, out 1
104	pop, out 2
105	out 2
106	out 2

IN label	action
100	out 2
101	out 2
102	out 1
103	pop, out 1
105	pop, out 2
106	out 2

IN label	action
100	out 3
101	pop, out 3
102	out 3
103	out 1
104	pop, out 1
106	pop, out 2

Fig. 6: Reference network scenario and testbed.

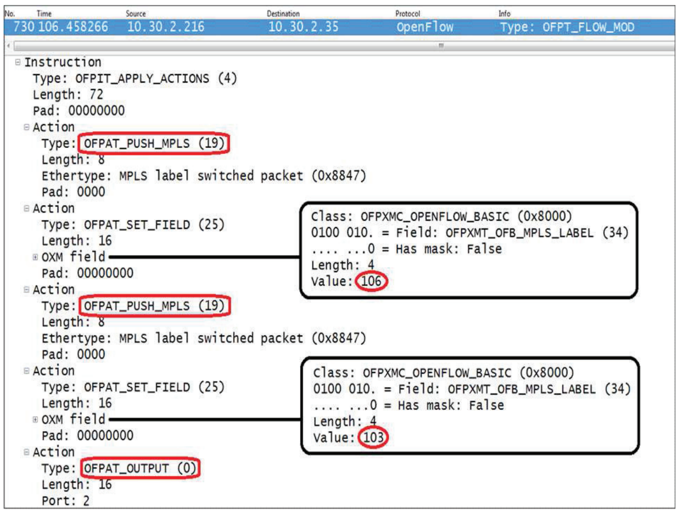
composed by the two segments $100 - 103$ and $104 - 106$.

B. Implemented data plane testbeds

The two multi-layer network testbeds reproducing the reference scenario shown in Fig. 6 mainly differ in the implementation of the PSC nodes and their related forwarding procedures.

In the testbed implementation utilized with the SDN-based SR Controller, the PSC nodes are implemented within Intel Core 4 servers (CPU 2.40 GHz, Kernel 2.6) equipped with 4 Gigabit Ethernet interfaces. The servers run OpenvSwitch version 2.61, supporting traditional MPLS-based forwarding. The considered OpenvSwitch software implementation has been modified by just changing the FLOW_MAX_MPLS_LABELS parameter, increasing the maximum supported segment list depth from 3 to 18 labels. In this case, the SR Controller is in charge of computing all the shortest paths and of configuring the required flow-entries in the switches. In particular, according to the considered network topology, the controller considers all the possible switch couples and for each of them it applies the following action: if the two nodes are adjacent, label popping is applied; otherwise the output forwarding port forwarding is set (i.e., label swapping with no changes in the label).

In the testbed implementation utilized with the PCE-based SR Controller, the PSC nodes are implemented by exploiting commercially available IP/MPLS routers. The routers support MPLS data plane forwarding and Open Shortest Path First (OSPF) routing protocol. However, no specific support of SR is provided (i.e., an operating system release supporting SR is not yet available at the time of writing). To overcome this issue and provide the required SR functionalities, a specifically designed external SR agent has been implemented. A dedicated SR agent is required for each IP/MPLS router deployed in the testbed. The agent north interface acts as a PCC, maintaining the PCEP connection with the SR Controller and possibly submitting new requests, whereas the south interface is in charge of configuring the co-located router. First, it retrieves the routing table from the running OSPF session. Then, it configures the shortest path entries in the local MPLS forwarding table by properly applying either label swapping (with same label value) or label popping. At the ingress routers, it also translates the NAI list enclosed in the `PCRep` message received from the PCE-based SR Controller in the list of SIDs to be configured in the local MPLS forwarding table. Such configuration in then enforced through label push operation(s).



```
No.  Time          Source       Destination  Protocol  Info
730 106.458266    10.30.2.216  10.30.2.35   OpenFlow  Type: OFPT_FLOW_MOD

⊟ Instruction
   Type: OFPIT_APPLY_ACTIONS (4)
   Length: 72
   Pad: 00000000
 ⊟ Action
    Type: OFPAT_PUSH_MPLS (19)
    Length: 8
    Ethertype: MPLS label switched packet (0x8847)
    Pad: 0000
 ⊟ Action
    Type: OFPAT_SET_FIELD (25)
    Length: 16
  ⊞ OXM field         Class: OFPXMC_OPENFLOW_BASIC (0x8000)
                      0100 010. = Field: OFPXMT_OFB_MPLS_LABEL (34)
                      .... ...0 = Has mask: False
                      Length: 4
                      Value: 106
    Pad: 00000000
 ⊟ Action
    Type: OFPAT_PUSH_MPLS (19)
    Length: 8
    Ethertype: MPLS label switched packet (0x8847)
    Pad: 0000
 ⊟ Action
    Type: OFPAT_SET_FIELD (25)
    Length: 16
  ⊞ OXM field         Class: OFPXMC_OPENFLOW_BASIC (0x8000)
                      0100 010. = Field: OFPXMT_OFB_MPLS_LABEL (34)
                      .... ...0 = Has mask: False
                      Length: 4
                      Value: 103
    Pad: 00000000
 ⊟ Action
    Type: OFPAT_OUTPUT (0)
    Length: 16
    Port: 2
```

Fig. 7: Wireshark snapshot of the communication between the SR controller and the edge PSC node.

VI. EXPERIMENTAL DEMONSTRATION

This section considers a specific use case to demonstrate the effectiveness of the considered implementations of the SR technology. To this extent, the controller includes a routing policy to discriminate between packet and optical resources. The applied policy is based on a bandwidth threshold (e.g., 40% of the PSC interface capacity): if the used bandwidth is below the acceptable threshold, the controller provides a routing preference for packet resources, otherwise for optical ones.

A first traffic request arrives from node 100 to node 106 with bandwidth below the considered threshold. As illustrated in Fig. 6, two possible paths can be considered by the SR controller: $\bar{p}_1$ through PSC nodes and $\bar{p}_2$ exploiting the optical bypass $101 - 105$. Given the considered routing policy, although $\bar{p}_2$ is the shortest path, $\bar{p}_1$ is selected for the considered request. To apply $\bar{p}_1$, the SR controller configures the forwarding table of the ingress node 100. Upon configuration,

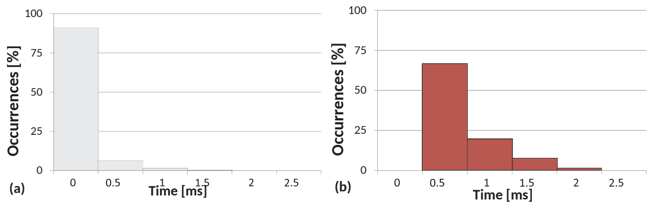


Fig. 8: Percentage distribution of the controller setup time [ms]: (a) 1 label, (b) 15-label deep stack.

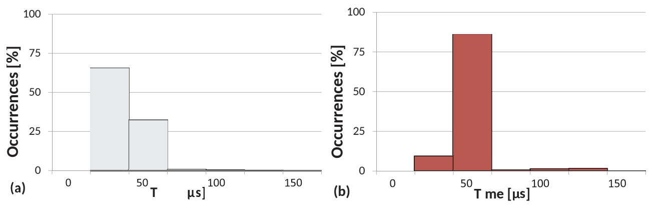


Fig. 10: Percentage distribution of the ingress switch setup time [us]: (a) 1 label, (b) 15-label deep stack.

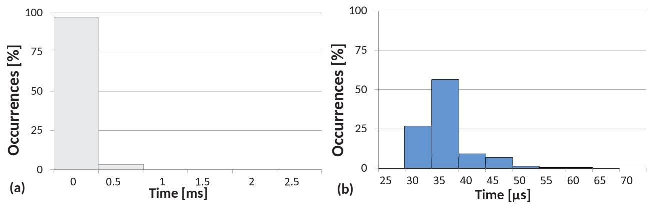


Fig. 9: (a): percentage distribution of PCE service time [ms]; (b) percentage distribution of algorithm computation time [us].

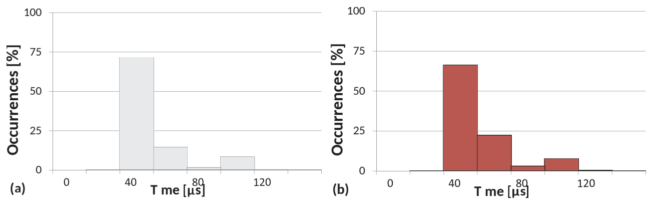


Fig. 11: Percentage distribution of the packet forwarding time [us]: (a) 1 label, (b) 15-label deep stack.

the segment list is enforced to the incoming packets, which are sent to the adjacent node 101. According to the included top label, packets are then forwarded to node 102 where penultimate hop popping is performed. Thus, following the new top label (i.e., 106) packets are forwarded toward the destination (with penultimate hop popping at node 105).

In case of SDN-based implementation the `OFP_FLOW_MOD` message shown in Fig. 7 is used to configure node 100 with required segment list, i.e. 106 as bottom label and 103 as top label. In case of PCE-based implementation a `PCRep` is used as shown in Fig. 5.

Now let's assume that the used bandwidth increases above the considered threshold due to a new traffic request from 100 to 106. The SR Controller performs a new constraint-based path computation selecting path $\bar{p}_2$, i.e. including the optical bypass. To apply the computed route, a new message is generated by the SR controller (i.e., `OFP_FLOW_MOD` or `PCRep` depending on the implementation) and sent to node 100. The message includes the recalculated segment list which encompasses just one label with value 106. Thus, node 100 updates the forwarding table and applies the new segment list. Packets are sent to 101 that, in this case, forwards the packets toward the optical bypass. Indeed the outport port 3 is indicated for packets arriving with label 106.

The described use case has been successfully experimented with both SR Controller implementations. No packet loss has been detected in any of the experiments.

VII. Scalability performance

To assess the scalability performance of the proposed SDN-based solution, a 10x10 Manhattan meshed network has been emulated. Traffic requests having different source-destination pairs have been considered. Specifically, two cases are here reported, providing statistics obtained by the repetition of 5000 experiments. In the first case, all the source-destination pairs having a unique shortest path are considered. For these paths, a segment list composed of a single label is sufficient to enforce the desired path. In the second case, all the paths requiring a 15-label deep segment list to be strictly identified among other equal cost paths are considered. Therefore, in this case, the edge node has to apply a stack of 15 labels to each forwarded packet.

Fig. 8 shows the distribution of the processing time required by the SDN-based SR controller to perform the required operations, i.e. path computation, segment list computation, and `OFP_FLOW_MOD` message generation. The comparison between the cases with 1 in Fig. 8(a) and 15 labels in Fig. 8(b) shows that the average processing time almost triples when deep label stacking has to be computed (average value from 273.7 μs for 1 label to 934.3 μs for 15 labels). This increase is mainly due to time required for path and label stacking computation.

Fig. 9 shows the results obtained with the PCE-based SR controller. Fig. 9(a) shows the distribution of PCE service time, computed as the time interval between the instant at which the `PCReq` message is received and the one at which the `PCRep` message is sent. In particular, this distribution has been obtained considering the testbed scenario in Fig. 6 where, according to the network status and bandwidth requirements, only segment lists composed by 1 or 2 labels are used. Comparing it with the SDN-based result with 1 label in the stack, Fig. 8(a), the two results are similar, considering that in both the SR Controller implementations the same path computation and segment list algorithms are adopted. Fig. 9(b) shows the distribution of segment list computation algorithm time, described in Sec. III-A. Also in this case the distribution has been obtained considering the testbed scenario Fig. 6 (i.e., maximum label stack dept of 2). The average algorithm time is 33.3 μs and all obtained samples are below 55 μs.

The processing time required by the edge PSC node to configure the forwarding table upon reception of the `OFP_FLOW_MOD` message has been also analyzed. Fig. 10 shows the distribution of the switch setup time. The switch setup time is computed as the difference between the time in which the switch receives the `OFP_FLOW_MOD` message and the time in which that flow-entry installation is completed. The edge node is the node requiring the highest switch setup time because it is in charge of configuring the longer segment list. The comparison between the cases with 1 label in Fig. 10(a) and 15 labels in Fig. 10(b) shows that the switch setup time slightly increases even when a long segment list needs to be configured (average value from 51.2 μs for 1 label to 58.7 μs for 15 labels). That is, no relevant delay is introduced in the control plane message elaboration and forwarding table configuration when SR is adopted, even with a deep label stack.

Fig. 11 shows the distribution of data packet forwarding time at the edge PSC node. It is the node requiring the highest forwarding time because it is in charge of applying the whole label stack. The forwarding time is computed as the difference between the time in which the IP packet is received at node 100 from host H1 and the time in which the packet exits the edge node with the proper segment list. The comparison between the cases with 1 label in Fig. 11(a) and 15 labels in Fig. 11(b) shows that no relevant forwarding latency is introduced even when deep label stacking is applied on the incoming packets (average value from 50.6 μs for 1 label to 53.1 μs for 15 labels).

VIII. Conclusions

Segment Routing (SR) technology has been implemented and successfully demonstrated in two different multi-layer network testbeds.

In the first testbed, a software defined networking (SDN) scenario is considered. Through a specifically designed OpenFlow message, packet nodes (i.e., OpenFlow switches) are configured to enforce the required segment list in accordance with the SR architecture.

In the second testbed, a novel Path Computation Element (PCE) scenario is considered. In this case, an enhanced PCEP message is introduced to enable the configuration at packet nodes of the proper segment list. A specifically designed SR agent has been implemented to enable the SR configuration of commercially available IP/MPLS routers.

Both implementations have been successfully utilized to demonstrate dynamic packet flow rerouting enabled by the enforcing of different segment list configurations at the ingress nodes. The two implementations rely on a novel path computation algorithm to determine both the strict route and the applied segment list. The algorithm has to be carefully designed in order to avoid scalability issues.

Experimental results show no packet loss during rerouting operation, which has been successfully performed without requiring the use of signaling protocols.

Scalability tests have shown that if an extremely deep segment list is applied, the time required to perform the overall flow configuration increases to an average value of about 1 ms, mainly due to path computation at the SR controller. However, no performance degradation has been experienced after flow configuration, i.e., the packet forwarding time is almost independent on the label stack depth.

Acknowledgment

This paper is an extended version of the work presented in [1]. Special acknowledgements to G. Bruno, J. Nijhof, and F. Lazzeri, Ericsson, Genova, Italy. This work was partially funded by the European Community's Seventh Framework Programme FP7-2007-2013 through the Integrated Project (IP) IDEALIST under grant agreement num. 317999.

References

[1] A. Sgambelluri, A. Giorgetti, F. L. Cugini, G. Bruno, and P. Castoldi, "First demonstration of sdn-based segment routing in multi-layer networks," in *Optical Fiber Communication Conference*, 2015, p. Th1.A.5.

[2] C. Filsfils *et al.*, "Segment routing architecture," *draft-filsfils-spring-segment-routing*, 2014.

[3] ——, "Segment routing with MPLS data plane," *draft-filsfils-spring-segment-routing-mpls*, 2014.

[4] D. Awduche *et al.*, "RSVP-TE: Extensions to RSVP for LSP tunnels," IETF RFC 3209, Dec. 2001.

[5] M. Vigoureux, B. Berde, L. Andersson, T. Cinkler, L. Levrau, M. Ondata, D. Colle, J. Fernandez-Palacios, and M. Jager, "Multilayer traffic engineering for gmpls-enabled networks," *Communications Magazine, IEEE*, vol. 43, no. 7, pp. 44–50, July 2005.

[6] A. Farrel and I. Bryskin, *GMPLS: Architecture and Applications*. The Morgan Kaufmann Series in Networking, 2005.

[7] "Segment routing." [Online]. Available: https://www.opennetworking.org/

[8] "Interoperability and feasibility showcase 2015 white paper." [Online]. Available: http://www.eantc.de/

[9] S. Bidkar, A. Gumaste, P. Ghodasara, A. Kushwaha, J. Wang, and A. Somani, "Scalable segment routing - a new paradigm for efficient service provider networking using carrier ethernet advances," *Optical Communications and Networking, IEEE/OSA Journal of*, vol. 7, no. 5, pp. 445–460, May 2015.

[10] D. Cai, A. Wielosz, and S. Wei, "Evolve carrier Ethernet architecture with SDN and segment routing," in *Proc. WoWMoM*, Jun. 2014.

[11] F. Lazzeri, G. Bruno, J. Nijhof, A. Giorgetti, and C. P., "Efficient label encoding in segment-routing enabled optical networks," in *Proc. ONDM*, May 2015.

[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," in *Proc. SIGCOMM*, Aug. 2008.

[13] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "OpenFlow-based segment protection in Ethernet networks," *J. Opt. Commun. Netw.*, vol. 5, no. 9, pp. 1066–1075, Sep. 2013.

[14] R. Casellas, R. Muñoz, R. Martínez, R. Vilalta, L. Liu, T. Tsuritani, I. Morita, V. Lopez, J. Fernández-Palacios *et al.*, "Sdn based provisioning orchestration of OpenFlow/GMPLS flexi-grid networks with a stateful hierarchical PCE," in *OFC Conf.*, 2014.

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

[16] A. Farrel and I. Bryskin, *GMPLS: Architecture and Applications*. Morgan Kauffmann, 2006.

[17] A. Bukva, R. Casellas, R. Martinez, and R. Munoz, "A dynamic path-computation algorithm for a gmpls-enabled multi-layer network," *J. Opt. Commun. Netw.*, vol. 4, no. 6, pp. 436–448, Jun. 2012.

[18] G. Swallow, "From tag switching to SDN and segment routing MPLS: an enduring architecture," in *Proc. MPLS SDN World Congress*, Mar. 2014.

[19] N. Sambo, A. Giorgetti, F. Cugini, N. Andriolli, L. Valcarenghi, and P. Castoldi, "Accounting for shared regenerators in gmpls-controlled translucent optical networks," *Lightwave Technology, Journal of*, vol. 27, no. 19, pp. 4338–4347, Oct 2009.

[20] A. Giorgetti, F. Cugini, N. Sambo, F. Paolucci, N. Andriolli, and P. Castoldi, "Path state-based update of pce traffic engineering database in wavelength switched optical networks," *Communications Letters, IEEE*, vol. 14, no. 6, pp. 575–577, June 2010.

[21] "Ryu controller." [Online]. Available: http://osrg.github.io/ryu/

[22] "OpenFlow switch specification 1.3.0," http://www.openflow.org/, Jun. 2012.

[23] S. Sivalaban, J. Medved, C. Filsfils, E. Crabbe, R. Raszuk, V. Lopez, and J. Tantsura, "Pcep extensions for segment routing," *draft-sivalaban-pce-segment-routing*, 2014.