SPRING
Internet-Draft
Intended status: Standards Track
Expires: April 9, 2018

F. Clad, Ed.
C. Filsfils
P. Camarillo
Cisco Systems, Inc.
D. Bernier
Bell Canada
B. Decraene
Orange
B. Peirens
Proximus
C. Yadlapalli
AT&T
X. Xu
Huawei
S. Salsano
Universita di Roma "Tor Vergata"
A. AbdelSalam
Gran Sasso Science Institute
G. Dawra
Cisco Systems, Inc.
October 6, 2017

Segment Routing for Service Chaining
draft-clad-spring-segment-routing-service-chaining-00

Abstract

This document defines data plane functionality required to implement service segments and achieve service chaining with MPLS and IPv6, as described in the Segment Routing architecture.

Table of Contents

1. Introduction

Segment Routing (SR) is an architecture based on the source routing paradigm that seeks the right balance between distributed intelligence and centralized programmability. SR can be used with an MPLS or an IPv6 data plane to steer packets through an ordered list of instructions, called segments. These segments may encode simple routing instructions for forwarding packets along a specific network path, or rich behaviors to support use-cases such as service chaining.

In the context of service chaining, each service, running either on a physical appliance or in a virtual environment, is associated with a segment, which can then be used in a segment list to steer packets through the service. Such service segments may be combined together in a segment list to achieve service chaining, but also with other types of segments as defined in [I-D.ietf-spring-segment-routing]. SR thus provides a fully integrated solution for service chaining, overlay and underlay optimization. Furthermore, the IPv6 dataplane natively supports metadata transportation as part of the SR information attached to the packets.

This document describes how SR enables service chaining in a simple and scalable manner, from the segment association to the service up to the traffic classification and steering into the service chain. Several SR proxy behaviors are also defined to support SR service chaining through legacy, SR-unaware, services in various circumstances.

The definition of control plane components, such as segment discovery and SR policy configuration, is outside the scope of this data plane document. These aspects will be defined in a dedicated document.

Familiarity with the following IETF documents is assumed:

- Segment Routing Architecture [I-D.ietf-spring-segment-routing]
- Segment Routing with MPLS data plane [I-D.ietf-spring-segment-routing-mpls]
- Segment Routing Header [I-D.ietf-6man-segment-routing-header]
- SRv6 Network Programming [I-D.filsfils-spring-srv6-network-programming]

2. Terminology

SR-aware service: Service fully capable of processing SR traffic

SR-unaware service: Service unable to process SR traffic or behaving incorrectly for such traffic

SR proxy: Proxy handling the SR processing on behalf of an SR-unaware service

Service Segment: Segment associated with a service, either directly or via an SR proxy

SR SC policy: SR policy, as defined in [I-D.filsfils-spring-segment-routing-policy], that includes at least one Service Segment. An SR SC policy may also contain other types of segments, such as VPN or TE segments.

SR policy head-end: SR node that classifies and steers traffic into an SR policy.

3. Classification and steering

Classification and steering mechanisms are defined in section 12 of [I-D.filsfils-spring-segment-routing-policy] and are independent from the purpose of the SR policy. From a headend perspective, there is no difference whether a policy contains IGP, BGP, peering, VPN and service segments, or any combination of these.

As documented in the above reference, traffic is classified when entering an SR domain. The SR policy head-end may, depending on its capabilities, classify the packets on a per-destination basis, via simple FIB entries, or apply more complex policy routing rules requiring to look deeper into the packet. These rules are expected to support basic policy routing such as 5-tuple matching. In addition, the IPv6 SRH tag field defined in [I-D.ietf-6man-segment-routing-header] can be used to identify and classify packets sharing the same set of properties. Classified traffic is then steered into the appropriate SR policy, which is associated with a weighted set of segment lists.

SR traffic can be re-classified by an SR endpoint along the original SR policy (e.g., DPI service) or a transit node intercepting the traffic. This node is the head-end of a new SR policy that is imposed onto the packet, either as a stack of MPLS labels or as an IPv6 and SRH encapsulation.

4. Services

A service may be a physical appliance running on dedicated hardware, a virtualized service inside an isolated environment such as a VM, container or namespace, or any process running on a compute element. Unless otherwise stated, this document does not make any assumption on the type or execution environment of a service.

SR enables service chaining by assigning a segment identifier, or SID, to each service and sequencing these service SIDs in a segment list. A service SID may be of local significance or directly reachable from anywhere in the routing domain. The latter is realized with SR-MPLS by assigning a SID from the global label block ([I-D.ietf-spring-segment-routing-mpls]), or with SRv6 by advertising the SID locator in the routing protocol ([I-D.filsfils-spring-srv6-network-programming]).

This document categorizes services in two types, depending on whether they are able to behave properly in the presence of SR information or not. These are respectively named SR-aware and SR-unaware services. An SR-aware service can process the SR information in the packets it receives. This means being able to identify the active segment as a local instruction and move forward in the segment list, but also that the service own behavior is not hindered due to the presence of SR information. For example, an SR-aware firewall filtering SRv6 traffic based on its final destination must retrieve that information from the last entry in the SRH rather than the Destination Address field of the IPv6 header. Any service that does not meet these criteria is considered as SR-unaware.

4.1. SR-aware services

An SR-aware service is associated with a locally instantiated service segment, which is used to steer traffic through it.

If the service is configured to intercept all the packets passing through the appliance, the underlying routing system only has to implement a default SR endpoint behavior (SR-MPLS node segment or SRv6 End function), and the corresponding SID will be used to steer traffic through the service.

If the service requires the packets to be directed to a specific virtual interface, networking queue or process, a dedicated SR behavior may be required to steer the packets to the appropriate location. The definition of such service-specific functions is out of the scope of this document.

An SRv6-aware service may also retrieve, store or modify information in the SRH TLVs.

4.2. SR-unaware services

An SR-unaware service is not able to process the SR information in the traffic that it receives. It may either drop the traffic or take erroneous decisions due to the unrecognized routing information. In order to include such services in an SR SC policy, it is thus required to remove the SR information before the service receives the packet, or to alter it in such a way that the service can correctly process the packet.

In this document, we define the concept of an SR proxy as an entity, separate from the service, that performs these modifications and handle the SR processing on behalf of a service. The SR proxy can run as a separate process on the service appliance, on a virtual switch or router on the compute node or on a remote host. In this document, we only assume that the proxy is connected to the service via a layer-2 link.

An SR-unaware service is associated with a service segment instantiated on the SR proxy, which is used to steer traffic through the service. Section 5 describes several SR proxy behaviors to handle the SR information under various circumstances.

5. SR proxy behaviors

This section describes several SR proxy behaviors designed to enable SR service chaining through SR-unaware services. A system implementing one of these functions may handle the SR processing on behalf of an SR-unaware service and allows the service to properly process the traffic that is steered through it.

A service may be located at any hop in an SR policy, including the last segment. However, the SR proxy behaviors defined in this section are dedicated to supporting SR-unaware services at intermediate hops in the segment list. In case an SR-unaware service is at the last segment, it is sufficient to ensure that the SR information is ignored (IPv6 routing extension header with Segments Left equal to 0) or removed before the packet reaches the service (MPLS PHP, SRv6 End.D or PSP).

As illustrated on Figure 1, the generic behavior of an SR proxy has two parts. The first part is in charge of passing traffic from the network to the service. It intercepts the SR traffic destined for the service via a locally instantiated service segment, modifies it in such a way that it appears as non-SR traffic to the service, then

sends it out on a given interface, IFACE-OUT, connected to the service. The second part receives the traffic coming back from the service on IFACE-IN, restores the SR information and forwards it according to the next segment in the list. Unless otherwise stated IFACE-OUT and IFACE-IN can represent the same interface.

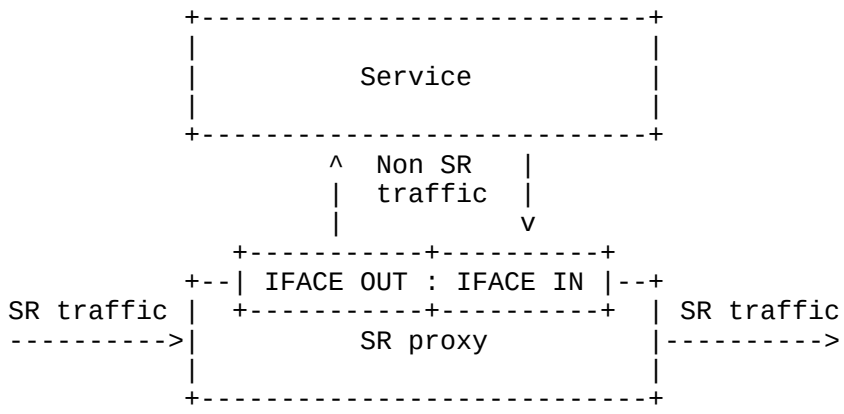

Figure 1: Generic SR proxy

In the next subsections, the following SR proxy mechanisms are defined:

- Static proxy
- Dynamic proxy
- Shared-memory proxy
- Masquerading proxy

Each mechanism has its own characteristics and constraints, which are summarized in the below table. It is up to the operator to select the best one based on the proxy node capabilities, the service behavior and the traffic type. It is also possible to use different proxy mechanisms within the same service chain.

		Static	Dynamic	Shared mem.	Masquerading
SR flavors	SR-MPLS	Y	Y	Y	-
	SRv6 insertion	P	P	P	Y
	SRv6 encapsulation	Y	Y	Y	-
Inner header	Ethernet	Y	Y	Y	-
	IPv4	Y	Y	Y	-
	IPv6	Y	Y	Y	-
Chain agnostic configuration		N	N	Y	Y
Transparent to chain changes		N	Y	Y	Y
Service support	DA modification	Y	Y	Y	NAT
	Payload modification	Y	Y	Y	Y
	Packet generation	Y	Y	cache	cache
	Packet deletion	Y	Y	Y	Y
	Transport endpoint	Y	Y	cache	cache

Figure 2: SR proxy summary

Note: The use of a shared memory proxy requires both the service and the proxy to be running on the same node.

5.1. Static SR proxy

The static proxy is an SR endpoint behavior for processing SR-MPLS or SRv6 encapsulated traffic on behalf of an SR-unaware service. This proxy thus receives SR traffic that is formed of an MPLS label stack or an IPv6 header on top of an inner packet, which can be Ethernet, IPv4 or IPv6.

A static SR proxy segment is associated with the following mandatory parameters:

- INNER-TYPE: Inner packet type
- S-ADDR: Ethernet or IP address of the service (only for inner type IPv4 and IPv6)
- IFACE-OUT: Local interface for sending traffic towards the service
- IFACE-IN: Local interface receiving the traffic coming back from the service
- CACHE: SR information to be attached on the traffic coming back from the service

A static SR proxy segment is thus defined for a specific service, inner packet type and cached SR information. It is also bound to a pair of directed interfaces on the proxy. These may be both directions of a single interface, or opposite directions of two different interfaces. The latter is recommended in case the service is to be used as part of a bi-directional SR SC policy. If the proxy and the service both support 802.1Q, IFACE-OUT and IFACE-IN can also represent sub-interfaces.

The first part of this behavior is triggered when the proxy node receives a packet whose active segment matches a segment associated with the static proxy behavior. It removes the SR information from the packet then sends it on a specific interface towards the associated service. This SR information corresponds to the full label stack for SR-MPLS or to the encapsulation IPv6 header with any attached extension header in the case of SRv6.

The second part is an inbound policy attached to the proxy interface receiving the traffic returning from the service, IFACE-IN. This policy attaches to the incoming traffic the cached SR information associated with the SR proxy segment. If the proxy segment uses the SR-MPLS data plane, CACHE contains a stack of labels to be pushed on top the packets. With the SRv6 data plane, CACHE is defined as a source address, an active segment and an optional SRH (tag, segments

left, segment list and metadata). The proxy encapsulates the packets with an IPv6 header that has the source address, the active segment as destination address and the SRH as a routing extension header. After the SR information has been attached, the packets are forwarded according to the active segment, which is represented by the top MPLS label or the IPv6 Destination Address.

In this scenario, there are no restrictions on the operations that can be performed by the service on the stream of packets. It may operate at all protocol layers, terminate transport layer connections, generate new packets and initiate transport layer connections. This behavior may also be used to integrate an IPv4-only service into an SRv6 policy. However, a static SR proxy segment can be used in only one service chain at a time. As opposed to most other segment types, a static SR proxy segment is bound to a unique list of segments, which represents a directed SR SC policy. This is due to the cached SR information being defined in the segment configuration. This limitation only prevents multiple segment lists from using the same static SR proxy segment at the same time, but a single segment list can be shared by any number of traffic flows. Besides, since the returning traffic from the service is re-classified based on the incoming interface, an interface can be used as receiving interface (IFACE-IN) only for a single SR proxy segment at a time. In the case of a bi-directional SR SC policy, a different SR proxy segment and receiving interface are required for the return direction.

5.1.1. SR-MPLS pseudocode

5.1.1.1. Static proxy for inner type Ethernet - MPLS L2 static proxy segment

Upon receiving an MPLS packet with top label L, where L is an MPLS L2 static proxy segment, a node N does:

```
1.   IF payload type is Ethernet THEN
2.       Pop all labels
3.       Forward the exposed frame on IFACE-OUT
4.   ELSE
5.       Drop the packet
```

Upon receiving on IFACE-IN an Ethernet frame with a destination address different than the interface address, a node N does:

```
1.   Push labels in CACHE on top of the frame Ethernet header
2.   Lookup the top label and proceed accordingly
```

The receiving interface must be configured in promiscuous mode in order to accept those Ethernet frames.

5.1.1.2. Static proxy for inner type IPv4 - MPLS IPv4 static proxy segment

Upon receiving an MPLS packet with top label L, where L is an MPLS IPv4 static proxy segment, a node N does:

```
1.  IF payload type is IPv4 THEN
2.      Pop all labels
3.      Forward the exposed packet on IFACE-OUT towards S-ADDR
4.  ELSE
5.      Drop the packet
```

Upon receiving a non link-local IPv4 packet on IFACE-IN, a node N does:

```
1.  Push labels in CACHE on top of the packet IPv4 header
2.  Decrement inner TTL and update checksum
3.  Lookup the top label and proceed accordingly
```

5.1.1.3. Static proxy for inner type IPv6 - MPLS IPv6 static proxy segment

Upon receiving an MPLS packet with top label L, where L is an MPLS IPv6 static proxy segment, a node N does:

```
1.  IF payload type is IPv6 THEN
2.      Pop all labels
3.      Forward the exposed packet on IFACE-OUT towards S-ADDR
4.  ELSE
5.      Drop the packet
```

Upon receiving a non link-local IPv6 packet on IFACE-IN, a node N does:

```
1.  Push labels in CACHE on top of the packet IPv6 header
2.  Decrement inner Hop Limit
3.  Lookup the top label and proceed accordingly
```

5.1.2. SRv6 pseudocode

5.1.2.1. Static proxy for inner type Ethernet - End.AS2

Upon receiving an IPv6 packet destined for S, where S is an End.AS2 SID, a node N does:

```
1.  IF ENH == 59 THEN                                          ;; Ref1
2.      Remove the (outer) IPv6 header and its extension headers
3.      Forward the exposed frame on IFACE-OUT
4.  ELSE
5.      Drop the packet
```

Ref1: 59 refers to "no next header" as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving on IFACE-IN an Ethernet frame with a destination address different than the interface address, a node N does:

```
1.  IF CACHE.SRH THEN                                          ;; Ref2
2.      Push CACHE.SRH on top of the existing Ethernet header
3.      Set NH value of the pushed SRH to 59
4.  Push outer IPv6 header with SA, DA and traffic class from CACHE
5.  Set outer payload length and flow label
6.  Set NH value to 43 if an SRH was added, or 59 otherwise
7.  Lookup outer DA in appropriate table and proceed accordingly
```

Ref2: CACHE.SRH represents the SRH defined in CACHE, if any, for the static SR proxy segment associated with IFACE-IN.

The receiving interface must be configured in promiscuous mode in order to accept those Ethernet frames.

5.1.2.2. Static proxy for inner type IPv4 - End.AS4

Upon receiving an IPv6 packet destined for S, where S is an End.AS4 SID, a node N does:

```
1.  IF ENH == 4 THEN                                           ;; Ref1
2.      Remove the (outer) IPv6 header and its extension headers
3.      Forward the exposed packet on IFACE-OUT towards S-ADDR
4.  ELSE
5.      Drop the packet
```

Ref1: 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving a non link-local IPv4 packet on IFACE-IN, a node N does:

```
1.  IF CACHE.SRH THEN                                          ;; Ref2
2.      Push CACHE.SRH on top of the existing IPv4 header
3.      Set NH value of the pushed SRH to 4
4.  Push outer IPv6 header with SA, DA and traffic class from CACHE
5.  Set outer payload length and flow label
6.  Set NH value to 43 if an SRH was added, or 4 otherwise
7.  Decrement inner TTL and update checksum
8.  Lookup outer DA in appropriate table and proceed accordingly
```

Ref2: CACHE.SRH represents the SRH defined in CACHE, if any, for the static SR proxy segment associated with IFACE-IN.

5.1.2.3. Static proxy for inner type IPv6 - End.AS6

Upon receiving an IPv6 packet destined for S, where S is an End.AS6 SID, a node N does:

```
1.  IF ENH == 41 THEN                                          ;; Ref1
2.      Remove the (outer) IPv6 header and its extension headers
3.      Forward the exposed packet on IFACE-OUT towards S-ADDR
4.  ELSE
5.      Drop the packet
```

Ref1: 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N does:

```
1.  IF CACHE.SRH THEN                                          ;; Ref2
2.      Push CACHE.SRH on top of the existing IPv6 header
3.      Set NH value of the pushed SRH to 41
4.  Push outer IPv6 header with SA, DA and traffic class from CACHE
5.  Set outer payload length and flow label
6.  Set NH value to 43 if an SRH was added, or 41 otherwise
7.  Decrement inner Hop Limit
8.  Lookup outer DA in appropriate table and proceed accordingly
```

Ref2: CACHE.SRH represents the SRH defined in CACHE, if any, for the static SR proxy segment associated with IFACE-IN.

5.2. Dynamic SR proxy

The dynamic proxy is an improvement over the static proxy that dynamically learns the SR information before removing it from the incoming traffic. The same information can then be re-attached to the traffic returning from the service. As opposed to the static SR proxy, no CACHE information needs to be configured. Instead, the

dynamic SR proxy relies on a local caching mechanism on the node instantiating this segment. Therefore, a dynamic proxy segment cannot be the last segment in an SR SC policy. As mentioned at the beginning of Section 5, a different SR behavior should be used if the service is meant to be the final destination of an SR SC policy.

Upon receiving a packet whose active segment matches a dynamic SR proxy function, the proxy node pops the top MPLS label or applies the SRv6 End behavior, then compares the updated SR information with the cache entry for the current segment. If the cache is empty or different, it is updated with the new SR information. The SR information is then removed and the inner packet is sent towards the service.

The cache entry is not mapped to any particular packet, but instead to an SR SC policy identified by the receiving interface (IFACE-IN). Any non-link-local IP packet or non-local Ethernet frame received on that interface will be re-encapsulated with the cached headers as described in Section 5.1. The service may thus drop, modify or generate new packets without affecting the proxy.

5.2.1. SR-MPLS pseudocode

The static proxy SR-MPLS pseudocode is augmented by inserting the following instructions between lines 1 and 2.

```
1.   IF top label S bit is 0 THEN
2.       Pop top label
3.       IF C(IFACE-IN) different from remaining labels THEN  ;; Ref1
4.           Copy all remaining labels into C(IFACE-IN)       ;; Ref2
5.   ELSE
6.       Drop the packet
```

Ref1: A TTL margin can be configured for the top label stack entry to prevent constant cache updates when multiple equal-cost paths with different hop counts are used towards the SR proxy node. In that case, a TTL difference smaller than the configured margin should not trigger a cache update (provided that the labels are the same).

Ref2: C(IFACE-IN) represents the cache entry associated to the dynamic SR proxy segment. It is identified with IFACE-IN in order to efficiently retrieve the right SR information when a packet arrives on this interface.

In addition, the inbound policy should check that C(IFACE-IN) has been defined before attempting to restore the MPLS label stack, and drop the packet otherwise.

5.2.2. SRv6 pseudocode

The static proxy SRv6 pseudocode is augmented by inserting the following instructions between lines 1 and 2.

```
1.   IF NH=SRH & SL > 0 THEN
2.       Decrement SL and update the IPv6 DA with SRH[SL]
3.       IF C(IFACE-IN) different from IPv6 encaps THEN      ;; Ref1
4.           Copy the IPv6 encaps into C(IFACE-IN)           ;; Ref2
5.   ELSE
6.       Drop the packet
```

Ref1: "IPv6 encaps" represents the IPv6 header and any attached extension header.

Ref2: C(IFACE-IN) represents the cache entry associated to the dynamic SR proxy segment. It is identified with IFACE-IN in order to efficiently retrieve the right SR information when a packet arrives on this interface.

In addition, the inbound policy should check that C(IFACE-IN) has been defined before attempting to restore the IPv6 encapsulation, and drop the packet otherwise.

5.3. Shared memory SR proxy

The shared memory proxy is an SR endpoint behavior for processing SR-MPLS or SRv6 encapsulated traffic on behalf of an SR-unaware service. This proxy behavior leverages a shared-memory interface with the service in order to hide the SR information from an SR-unaware service while keeping it attached to the packet. We assume in this case that the proxy and the service are running on the same compute node. A typical scenario is an SR-capable vrouter running on a container host and forwarding traffic to virtual services isolated within their respective container.

More details will be added in a future revision of this document.

5.4. Masquerading SR proxy

The masquerading proxy is an SR endpoint behavior for processing SRv6 traffic on behalf of an SR-unaware service. This proxy thus receives SR traffic that is formed of an IPv6 header and an SRH on top of an inner payload. The masquerading behavior is independent from the inner payload type. Hence, the inner payload can be of any type but it is usually expected to be a transport layer packet, such as TCP or UDP.

A masquerading SR proxy segment is associated with the following mandatory parameters:

- S-ADDR: Ethernet or IPv6 address of the service
- IFACE-OUT: Local interface for sending traffic towards the service
- IFACE-IN: Local interface receiving the traffic coming back from the service

A masquerading SR proxy segment is thus defined for a specific service and bound to a pair of directed interfaces or sub-interfaces on the proxy. As opposed to the static and dynamic SR proxies, a masquerading segment can be present at the same time in any number of SR SC policies and the same interfaces can be bound to multiple masquerading proxy segments. The only restriction is that a masquerading proxy segment cannot be the last segment in an SR SC policy.

The first part of the masquerading behavior is triggered when the proxy node receives an IPv6 packet whose Destination Address matches a masquerading proxy segment. The proxy inspects the IPv6 extension headers and substitutes the Destination Address with the last segment in the SRH attached to the IPv6 header, which represents the final destination of the IPv6 packet. The packet is then sent out towards the service.

The service receives an IPv6 packet whose source and destination addresses are respectively the original source and final destination. It does not attempt to inspect the SRH, as RFC2460 specifies that routing extension headers are not examined or processed by transit nodes. Instead, the service simply forwards the packet based on its current Destination Address. In this scenario, we assume that the service can only inspect, drop or perform limited changes to the packets. For example, Intrusion Detection Systems, Deep Packet Inspectors and non-NAT Firewalls are among the services that can be supported by a masquerading SR proxy. Variants of the masquerading behavior are defined in Section 5.4.2 and Section 5.4.3 to support a wider range of services.

The second part of the masquerading behavior, also called de-masquerading, is an inbound policy attached to the proxy interface receiving the traffic returning from the service, IFACE-IN. This policy inspects the incoming traffic and triggers a regular SRv6 endpoint processing (End) on any IPv6 packet that contains an SRH. This processing occurs before any lookup on the packet Destination Address is performed and it is sufficient to restore the right active segment as the Destination Address of the IPv6 packet.

5.4.1. SRv6 masquerading proxy pseudocode - End.AM

Masquerading: Upon receiving a packet destined for S, where S is an End.AM SID, a node N processes it as follows.

```
1.   IF NH=SRH & SL > 0 THEN
2.       Update the IPv6 DA with SRH[0]
3.       Forward the packet on IFACE-OUT
4.   ELSE
5.       Drop the packet
```

De-masquerading: Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N processes it as follows.

```
1.   IF NH=SRH & SL > 0 THEN
2.       Decrement SL
3.       Update the IPv6 DA with SRH[SL]                    ;; Ref1
4.       Lookup DA in appropriate table and proceed accordingly
```

Ref2: This pseudocode can be augmented to support the Penultimate Segment Popping (PSP) endpoint flavor. The exact pseudocode modification are provided in [I-D.filsfils-spring-srv6-network-programming].

5.4.2. Variant 1: NAT

Services modifying the destination address in the packets they process, such as NATs, can be supported by a masquerading proxy with the following modification to the de-masquerading pseudocode.

De-masquerading - NAT: Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N processes it as follows.

```
1.   IF NH=SRH & SL > 0 THEN
2.       Update SRH[0] with the IPv6 DA
3.       Decrement SL
4.       Update the IPv6 DA with SRH[SL]
5.       Lookup DA in appropriate table and proceed accordingly
```

5.4.3. Variant 2: Caching

Services generating packets or acting as endpoints for transport connections can be supported by adding a dynamic caching mechanism similar to the one described in Section 5.2.

More details will be added in a future revision of this document.

6. Illustrations

We consider the network represented in Figure 3 where:

- A and B are two end hosts using IPv4
- B advertises the prefix 20.0.0.0/8
- 1 to 6 are physical or virtual routers supporting IPv6 and segment routing
- S1 is an SR-aware firewall service
- S2 is an SR-unaware IPS service

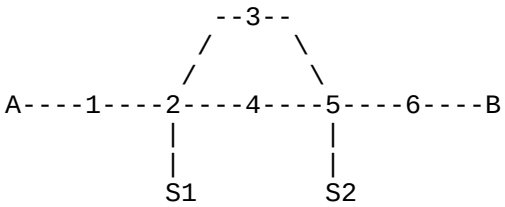

Figure 3: Network with services

All links are configured with an IGP weight of 10 except link 2-3 that is set to 20.

We assume that the path 2-3-5 has a lower latency than 2-4-5.

Nodes 1 to 6 each advertise in the IGP an IPv6 prefix Ck::/64, where k represents the node identifier.

Nodes 1 to 6 are each configured with an SRv6 End segment Ck::/128, where k represents the node identifier.

Node S1 is configured with an SRv6 SID CF1::/128 such that packets arriving at S1 with the Destination Address CF1:: are processed by the service. This SID is either advertised by S1, if it participates in the IGP, or by node 2 on behalf of S1.

Node 5 is also configured with an SRv6 dynamic proxy segments (End.AD) C5::AD:F2 for S2.

Node 6 is also configured with an SRv6 End.DX4 segment C6::D4:B decapsulating the SRv6 and sending the inner IPv4 packets towards D.

Via BGP signaling or an SDN controller, node 1 is programmed with a route 20.0.0.0/8 via C6::D4:B and a color/community requiring low latency and services S1 and S2.

Node 1 either locally computes the path to the egress node or delegates the computation to a PCE. As a result, the SRv6 encapsulation policy < CF1::, C3::, C5::AD:F2, C6::D4:B > is associated with the route 20.0.0.0/8 on node 1.

Upon receiving a packet P from node A and destined to 20.20.20.20, node 1 finds the above table entry and pushes an outer IPv6 header with (SA = C1::, DA = CF1::, NH = SRH) followed by an SRH (C6::D4:B, C5::AD:F2, C3::, CF1::; SL = 3; NH = IPv4). Node 1 then forwards the packet to the first destination address CF1::.

Node 2 forwards P along the shortest path to S1, based on the IPv6 destination address CF1::.

When S1 receives the packet, it identifies a locally instantiated SID and applies the firewall filtering rules. If the packet is not dropped, the SL value is decremented and the DA is updated to the next segment C3::. S1 then sends back to node 2 the packet P with (SA = C1::, DA = C3::, NH = SRH) (C6::D4:B, C5::AD:F2, C3::, CF1::; SL = 2; NH = IPv4).

Node 2 forwards P along the shortest path to node 3, based on the IPv6 destination address C3::.

When 3 receives the packet, 3 matches the DA in its local SID table and finds the bound End function. It thus decrements the SL value and updates the DA to the next segment: C5::AD:F2. Node 3 then forwards packet P with (SA = C1::, DA = C5::AD:F2, NH = SRH) (C6::D4:B, C5::AD:F2, C3::, CF1::; SL = 1; NH = IPv4) towards node 5.

When 5 receives the packet, 5 matches the DA in its local SID table and finds the bound function End.AD(S2). It thus performs the End function (decrement SL and update DA), caches and removes the outer IPv6 header and the SRH, then forwards the inner IPv4 packet towards S2.

S2 receives a regular IPv4 packet headed to 20.20.20.20. It applies the IPS rules and forwards the packet back to node 5.

When 5 receives the packet on the returning interface (IFACE-IN) for S2, 5 retrieves the corresponding cache entry and pushes the updated IPv6 header and SRH. It then forwards P with (SA = C1::, DA = C6:D4:B, NH = SRH) (C6::D4:B, C5::AD:F2, C3::, CF1::; SL = 0; NH = IPv4) to node 6.

When 6 receives the packet, 6 matches the DA in its local SID table and finds the bound function End.DX4. It thus removes the outer IPv6 header and forwards the inner IPv4 packet to node B.

7. Metadata

7.1. MPLS data plane

The MPLS data plane does not provide any native mechanism to attach metadata to a packet.

Workarounds to carry metadata in an SR-MPLS context will be discussed in a future version of this document.

7.2. IPv6 - SRH TLV objects

The IPv6 SRH TLV objects are designed to carry all sorts of metadata. In particular, [I-D.ietf-6man-segment-routing-header] defines the NSH carrier TLV as a container for NSH metadata.

TLV objects can be imposed by the ingress edge router that steers the traffic into the SR SC policy.

An SR-aware service may impose, modify or remove any TLV object attached to the first SRH, either by directly modifying the packet headers or via a control channel between the service and its forwarding plane.

An SR-aware service that re-classifies the traffic and steers it into a new SR SC policy (e.g. DPI) may attach any TLV object to the new SRH.

Metadata imposition and handling will be further discussed in a future version of this document.

7.3. IPv6 - SRH tag

The SRH tag identifies a packet as part of a group or class of packets [I-D.ietf-6man-segment-routing-header].

In a service chaining context, this field can be used as a simple man’s metadata to encode additional information in the SRH.

8. Implementation status

The static SR proxy is available for SR-MPLS and SRv6 on various Cisco hardware and software platforms. Furthermore, the following proxies are available on open-source software.

		VPP	Linux
MPLS	Static proxy	Available	In progress
	Dynamic proxy	In progress	In progress
	Shared memory proxy	In progress	In progress
SRv6	Static proxy	Available	In progress
	Dynamic proxy - Inner type Ethernet	In progress	In progress
	Dynamic proxy - Inner type IPv4	Available	Available
	Dynamic proxy - Inner type IPv6	Available	Available
	Shared memory proxy	In progress	In progress
	Masquerading proxy	Available	Available
	Masquerading proxy - NAT variant	In progress	In progress
	Masquerading proxy - Cache variant	In progress	In progress

Open-source implementation status table

9. Relationship with RFC 7665

The Segment Routing solution addresses a wider problem that covers both topological and service chaining policies. The topological and service instructions can be either deployed in isolation or in combination. SR has thus a wider applicability than the architecture defined in [RFC7665]. Furthermore, the inherent property of SR is a stateless network fabric. In SR, there is no state within the fabric to recognize a flow and associate it with a policy. State is only present at the ingress edge of the SR domain, where the policy is encoded into the packets. This is completely different from NSH that relies on state configured at every hop of the service chain.

Hence, there is no linkage between this document and [RFC7665].

10. IANA Considerations

This document has no actions for IANA.

11. Security Considerations

The security requirements and mechanisms described in [I-D.ietf-spring-segment-routing] and [I-D.ietf-6man-segment-routing-header] also apply to this document. Additional considerations will be discussed in future versions of the document.

12. Acknowledgements

TBD.

13. Contributors

Jisu Bhattacharya substantially contributed to the content of this document.

14. References

14.1. Normative References

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-12 (work in progress), June 2017.

14.2. Informative References

[I-D.filsfils-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Raza, K., Liste, J., Clad, F., Lin, S., bogdanov@google.com, b., Horneffer, M., Steinberg, D., Decraene, B., and S. Litkowski, "Segment Routing Policy for Traffic Engineering", draft-filsfils-spring-segment-routing-policy-01 (work in progress), July 2017.

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Matsushima, S., Lebrun, D., Decraene, B., Peirens, B., Salsano, S., Naik, G., Elmalky, H., Jonnalagadda, P., Sharif, M., Ayyangar, A., Mynam, S., Henderickx, W., Bashandy, A., Raza, K., Dukes, D., Clad, F., and P. Camarillo, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-01 (work in progress), June 2017.

[I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Raza, K., Leddy, J., Field, B., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-07 (work in progress), July 2017.

[I-D.ietf-spring-segment-routing-mpls]
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-10 (work in progress), June 2017.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

Authors' Addresses

Francois Clad (editor)
Cisco Systems, Inc.
France

Email: fclad@cisco.com

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Pablo Camarillo Garvia
Cisco Systems, Inc.
Spain

Email: pcamaril@cisco.com

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Bart Peirens
Proximus
Belgium

Email: bart.peirens@proximus.com

Chaitanya Yadlapalli
AT&T
USA

Email: cy098d@att.com

Xiaohu Xu
Huawei

Email: xuxiaohu@huawei.com

Stefano Salsano
Universita di Roma "Tor Vergata"
Italy

Email: stefano.salsano@uniroma2.it

Ahmed AbdelSalam
Gran Sasso Science Institute
Italy

Email: ahmed.abdelsalam@gssi.it

Gaurav Dawra
Cisco Systems, Inc.
USA

Email: gdawra@cisco.com