Network Programming with SRv6

Hernán Contreras G.
Technical Solution Architect, Cisco Systems
LACNOG - September 2018
Agenda

• Introduction to Segment Routing
• Introduction to SRv6
• SRv6 Network Programming
• SRv6 NP Use Cases
• SRv6 Interoperability and Demos
• References
Segment Routing
Segment Routing – Technical view

Path expressed in the packet

Data Plane

- MPLS (segment labels)
- IPv6 (+ SR extension header)

Control Plane

- Routing protocols with extensions (IS-IS, OSPF, BGP)
- SDN controller (BGP LS, PCEP, NETCONF/YANG)

Paths options

- Dynamic (STP computation)
- Explicit (expressed in the packet)

Data

Dynamic path

Explicit path
IETF

- Strong commitment for standardization and multi-vendor support
- SPRING Working-Group (started Nov 2013)
- All key documents are WG-status
- Over 25 drafts maintained by SR team
  - Over 50% are WG status
  - Over 75% have a Cisco implementation
- Several interop reports are available
- First RFC document – RFC 7855 (May 2016)
- RFC 8402 – Segment Routing Architecture

www.segment-routing.net
tools.ietf.org/wg/spring/

- Technology and Problem Statement
  - Architecture (draft)
  - Problem Statement
    - Generic (draft)
    - Resiliency (draft)
    - IPv6 (draft)
    - OAM (draft)
  - Applicability
    - SR Illustration to problem statement (draft)
    - Centralized Egress Peer Engineering (draft)

- Protocol Extension
  - IS-IS extension for SR (draft)
  - OSPF extension for SR (draft)
  - OSPFV3 extension for SR (draft)
  - BGP-LS extension for SR (draft)
  - BGP-LS extension for SR EPE use-case (draft)
  - PCEP extension for SR (SR ext, setup method)

- EBR
  - Topology-Independent LFA FRR with SR (draft)

- MPLS instantiation of Segment Routing
  - MPLS support for SR (draft)
  - SR/LDP Interaction and Interworking (draft)

- IPv6 instantiation of Segment Routing
  - IPv6 SR routing extension header (draft)
  - IPv6 use-cases (draft)

- QoS
  - SRV6SP Ping (draft)
  - OAM (draft)
A Standardized Technology
No Vendor Lock-In

• IETF Standardization in SPRING working group
• Protocol extensions progressing in multiple groups
  • IS-IS
  • OSPF
  • PCE
  • IDR
  • 6MAN
• Broad vendor and customer support
• 50% Draft-based working implementations
• 25% under development

<table>
<thead>
<tr>
<th>Sample IETF Documents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Statement and Requirements (RFC 7855)</td>
</tr>
<tr>
<td>Segment Routing Architecture (RFC 8402)</td>
</tr>
<tr>
<td>IPv6 SPRING Use Cases (RFC 8354)</td>
</tr>
<tr>
<td>Segment Routing with MPLS data plane</td>
</tr>
<tr>
<td>(draft-ietf-spring-segment-routing-mpls)</td>
</tr>
<tr>
<td>Topology Independent Fast Reroute using Segment Routing</td>
</tr>
<tr>
<td>(draft-francois-rtwg-segment-routing-ti-lfa)</td>
</tr>
<tr>
<td>IS-IS Extensions for Segment Routing (draft-ietf-isis-segment-routing-extensions)</td>
</tr>
<tr>
<td>OSPF Extensions for Segment Routing (draft-ietf-ospf-segment-routing-extensions)</td>
</tr>
<tr>
<td>PCEP Extensions for Segment Routing (draft-ietf-pce-segment-routing)</td>
</tr>
</tbody>
</table>

Around 40 IETF drafts in progress
SR = near-stateless traffic engineering

Packet

Explicit path “encoded” as Segment List

Application x

© 2017 Cisco and/or its affiliates. All rights reserved. Cisco Confidential
SR Segments

IGP Prefix Segment
- Signaled by ISIS/OSPF
- Minor extensions to the existing link-state routing protocols (OSPF and IS-IS)
- Shortest-path to the IGP prefix
- Global in SR domain
- SRGB + Index => 16000+5 = 16005

IGP Adjacency Segment
- Signaled by ISIS/OSPF
- Minor extensions to the existing link-state routing protocols (OSPF and IS-IS)
- Forward on the IGP adjacency
- Local
- Automatically allocated by the router
Segment Routing

- **Source Routing**
  - the source chooses a path and encodes it in the packet header as an ordered list of segments
  - the rest of the network executes the encoded instructions

The flow state is in the header, not in the network.

**Segment**: an identifier for any type of instruction (forwarding or service)

- **Forwarding Plane**:
  - **MPLS**: an ordered list of segments is represented as a stack of labels
  - **IPv6**: an ordered list of segments is encoded in a routing extension header
Segment Routing applies to both IPv6 and MPLS dataplanes. Difference is in the bits encoded in the packet not in the architecture.

**MPLS**: an ordered list of segments is represented as a stack of labels. Segment Routing re-uses MPLS data plane without any change. Segment represented as MPLS label. Applicable to IPv4 and IPv6 address families defined in `ietf-spring-segment-routing-mpls`.

**IPv6**: an ordered list of segments is encoded in a routing extension header.
SRv6 Header

draft-ietf-6man-segment-routing-header-14

This document describes the Segment Routing Extension Header and how it is used by Segment Routing capable nodes.
IPv6 SR Header

- IPv6 header
  - Next header field: 43 \(\rightarrow\) **Routing**
- IPv6 Routing extension header
  - Generic header format defined in RFC 8200
    - Next Header: IPv4, TCP, UDP, ...
    - Hdr Ext Len: Any IPv6 device can skip this header
    - Segments Left: Ignore extension header if equal to 0
  - Specific data depends on Routing Type field:
    - 0 - **Source Route** (deprecated since 2007)
    - 1 - **Nimrod** (deprecated since 2009)
    - 2 - Mobility (RFC 6275)
    - 3 - RPL Source Route (RFC 6554)
    - 4 - **Segment Routing** (suggested value 4)

The Routing header is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet's destination.

https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml
IPv6 SR Header

- Each segment is an IPv6 address
- Segments are encoded in reverse order
  - Last segment index is 0
  - First segment index is **First Segment**
  - Active segment index is **Segments Left**
- Active Segment is copied in the Destination Address field of the IP header
- Additional data can be stored in TLVs
  - Security (HMAC), NFV metadata, …
SRv6 Source Node

- Source node is SR-capable
- SR Header (SRH) is created with
  - Segment list in reversed order of the path
    - Segment List [0] is the LAST segment
    - Segment List [n-1] is the FIRST segment
  - Segments Left is set to n – 1
  - First Segment is set to n – 1
- IP DA is set to the first segment
- Packet is send according to the IP DA
  - Normal IPv6 forwarding
Non-SR Transit Node

Enabling SR-IPv6, means that ONLY the nodes that have to process the packet header must have SR-IPv6 dataplane support.

All other nodes in the infrastructure are just plain IPv6 nodes.

- Plain IPv6 forwarding
- Solely based on IPv6 DA
- No SRH inspection or update
SR Segment Endpoints

- SR Endpoints: SR-capable nodes whose address is in the IP DA
- SR Endpoints inspect the SRH and do:
  - IF Segments Left > 0, THEN
  - Decrement Segments Left ( -1 )
  - Update DA with Segment List [ Segments Left ]
  - Forward according to the new IP DA
SR Segment Endpoints

- SR Endpoints: SR-capable nodes whose address is in the IP DA
- SR Endpoints inspect the SRH and do:
  - IF Segments Left > 0, THEN
    - Decrement Segments Left (-1)
    - Update DA with Segment List [Segments Left]
    - Forward according to the new IP DA
  - ELSE (Segments Left = 0)
    - Remove the IP and SR header
    - Process the payload:
      - Inner IP: Lookup DA and forward
      - TCP / UDP: Send to socket

Standard IPv6 processing
The final destination does not have to be SR-capable.
SRv6 Network Programming

- SRv6 looks similar to SR MPLS, as in both cases you have a list of Segment IDs used to direct traffic over a specific path through the network
  - the exception of the Segment ID size – 128-bits for SRv6 compared to 32-bits for SR MPLS,
- That increase in size on SID opens the door to an entirely new paradigm: “Network Programming.”
  - With 128 bits, 4 times 32 bits, you can pack more than mere IP addresses into a Segment ID and hence go beyond routing purposes.
- 128-bits Segment IDs can be used and allocated for different purposes, for example:
  - The first 64 bits can be used to direct traffic to a specific node in the network – the “main body” of the program
  - The next 32 bits can be used to enforce some actions on the traffic – the “function” part
  - The remaining 32 bits can be used to pass some additional information – the “argument” part

<table>
<thead>
<tr>
<th>Locator</th>
<th>Function</th>
</tr>
</thead>
</table>
| 128-bit SRv6 SID | Locator: routed to the node performing the function
|              | Function: any possible function
|              | Flexible bit-length selection |
SRv6 Network Programming

SRv6 goes one step further by enabling the infrastructure to perform some actions on the applications: you have now the ability to code directly into each packet header where the traffic should be sent and how the traffic should be treated.

Locator: routed to the node performing the function

Function: any possible function either local to NPU or app in VM/Container

Arguments: optional argument bits to be used only by that SID

Locator: routed to the node performing the function

Function: any possible function either local to NPU or app in VM/Container

Arguments: optional argument bits to be used only by that SID

Segment format

<table>
<thead>
<tr>
<th>Locator</th>
<th>Function</th>
</tr>
</thead>
</table>

- SRv6 SIDs are 128-bit addresses
  - **Locator**: most significant bits are used to **route** the segment to its **parent node**
  - **Function**: least significant bits identify the **action** to be performed on the **parent node**
    - **Argument** [optional]: Last bits can be used as a local function argument

- Flexible bit-length allocation
  - Segment format is **local knowledge** on the parent node

- SIDs have to be **specifically enabled** as such on their parent node
  - A local address **is not** by default a local SID
  - A local SID does not have to be associated with an interface
## Endpoint behaviors specs summary

<table>
<thead>
<tr>
<th>Codename</th>
<th>Behavior</th>
<th>[PSP/USP flavors]</th>
</tr>
</thead>
<tbody>
<tr>
<td>End</td>
<td>Endpoint</td>
<td></td>
</tr>
<tr>
<td>End.X</td>
<td>Endpoint with Layer-3 cross-connect</td>
<td>[PSP/USP flavors]</td>
</tr>
<tr>
<td>End.B6</td>
<td>Endpoint bound to an SRv6 policy</td>
<td></td>
</tr>
<tr>
<td>End.B6.Encaps</td>
<td>Endpoint bound to an SRv6 Encapsulation policy</td>
<td></td>
</tr>
<tr>
<td>End.DX6</td>
<td>Endpoint with decapsulation and IPv6 cross-connect</td>
<td>(per-CE VPN label)</td>
</tr>
<tr>
<td>End.DX4</td>
<td>Endpoint with decapsulation and IPv4 cross-connect</td>
<td>(per-CE VPN label)</td>
</tr>
<tr>
<td>End.DT6</td>
<td>Endpoint with decapsulation and specific IPv6 table lookup</td>
<td>(per-VRF VPN label)</td>
</tr>
<tr>
<td>End.DT4</td>
<td>Endpoint with decapsulation and specific IPv4 table lookup</td>
<td>(per-VRF VPN label)</td>
</tr>
<tr>
<td>End.DX2</td>
<td>Endpoint with decapsulation and Layer-2 cross-connect</td>
<td></td>
</tr>
</tbody>
</table>
END SIDs

- **END function**
  - shortest-path to SID’s endpoint
  - endpoint updates DA with next SID
  - endpoints forwards according to updated DA
  - 0:0:0:0:1 is commonly used for an END function

Node 2 advertises prefix A2::/64 (A2::/64 is the SID **locator**)

```
1. IF NH=SRH and SL > 0
2. decrement SL
3. update the IPv6 DA with SRH[SL]
4. FIB lookup on updated DA
5. forward accordingly to the matched entry
6. ELSE
7. drop the packet
```
END.X SID

- END.X function
  - shortest-path to SID’s endpoint
  - endpoint updates DA with next SID
  - endpoint forwards to interface associated with SID
  - 0:0:0:0:Ck is commonly used for an END.X function where k identifies the interface
    - e.g. 0:0:0:0:C4 is the END.X of node 2 associated with the interface to 4

On Node 2, the *endpoint xconnect* behavior for link (2, 4) is associated with ID C4
The SID corresponding to *endpoint xconnect-(2,4)* behavior on node 2 is **A2::C4**

My LocalSID table: A2::C4 -> End.X {OIF=4; NH=2001::1}
End.DX6: Endpoint with decapsulation and IPv6 xconnect

- Equivalent to per-CE VPN label in MPLS
- If (SL == 0 & NH == 41“IPv6”)
  - Pop the (outer) IPv6 header and its extension headers
  - **Forward on the outgoing interface associated with the localSID**
- My LocalSID table: A2::DC6:1 -> End.DX6 {OIF=1; NH=2001::1}
END.DX6 Use Case: BGP VPNv6

Site A

Site B

1

2

3

Lo0 A:1::

Lo0 A:3::

END.DX6-A:3:1:0:1::

VRF:Enterprise100

BGP Signaling VPN SID
Session: BGP VPNv6
NH: A:3::
VPN SID: A:3:1:0:1::

END.DX6

iBGP-VPNv6
END.DX6 Use Case: BGP VPNv6

**BGP installs Prefix into RIB**
- Prefix: E:B::/64
- VPN SID: A:3:1:0:1::

Site A

Best Effort Traffic
- SA:E:A::A
- DA:E:B::B
- NH:UDP
- UDP Header/Data

Site B

Prefix: E:B::/64
- VPN SID: A:3:1:0:1::
- VRF:Enterprise100

iBGP-VPNv6
END.DX6 Use Case: BGP VPNv6
END.DX6 Use Case: BGP VPNv6

Site A
Best Effort Traffic
- SA:E:A::A
- DA:E:B::B
- NH:UDP
- UDP Header/Data

Site B
- Locator: END.DX6-A:3:1:0:1::
- Function: VRF:Enterprise100
- SA:E:A::A
- DA:E:B::B
- NH:UDP
- UDP Header/Data

© 2016 Cisco and/or its affiliates. All rights reserved. Cisco Confidential
END.DX6 Use Case: BGP VPNv6

Site A

Best Effort Traffic

Lo0 A:1::

Site B

Locator

Function

END.DX6-A:3:1:0:1::
VRF:Enterprise100
SRv6-based VPN (SRv6-VPN) refers to the creation of VPN between PE's leveraging the SRv6 dataplane and more specifically the END.DT* (crossconnect to a VRF) and END.DX* (crossconnect to a nexthop) functions defined in the SRv6 network programming document.
End.DX4: Endpoint with decapsulation and IPv4 xconnect

- Equivalent to per-CE VPN label in MPLS
- If (SL == 0 & NH == 4"IPv4")
  - Pop the (outer) IPv6 header and its extension headers
  - Forward on the outgoing interface associated with the localSID
- My LocalSID table: A2::DC4 -> End.DX4 {OIF=1; NH=2.2.2.0}
VPNv4 – Basic SRv6 VPN – Control Plane

- eBGP AFI:1 -IPv4 SAFI:1 NLRI:4.0.0.0/8 NH:3.3.3.2
- iBGP AFI:1 -IPv4 SAFI:128 NLRI:4.0.0.0/8 NH:A2::1 Label:ImplNull SID: A2::C4
- eBGP AFI:1 -IPv4 SAFI:1 NLRI:4.0.0.0/8 NH:4.4.4.1

3.0.0.0/8
A1::1 A1::C3 end.DX4
3.3.3.1
3.3.3.2
1

4.0.0.0/8
A2::1 A2::C4 end.DX4
4.4.4.1
4.4.4.2
2

Sales & Partner Training
Worldwide Sales Strategy & Operations

© 2016 Cisco and/or its affiliates. All rights reserved. Cisco Confidential
VPNv4 – Basic SRv6 VPN -Data Plane
Functions Defined in Net Programming

- **End** Endpoint function The SRv6 instantiation of a prefix SID
- **End.X** Endpoint function with Layer-3 cross-connect The SRv6 instantiation of a Adj SID
- **End.T** Endpoint function with specific IPv6 table lookup
- **End.DX2** Endpoint with decapsulation and Layer-2 cross-connect L2VPN use-case
- **End.DX2V** Endpoint with decapsulation and VLAN L2 table lookup EVPN Flexible cross-connect use-cases
- **End.DT2U** Endpoint with decaps and unicast MAC L2 table lookup EVPN Bridging unicast use-cases
- **End.DT2M** Endpoint with decapsulation and L2 table flooding EVPN Bridging BUM use-cases with ESI filtering
- **End.DX6** Endpoint with decapsulation and IPv6 cross-connect IPv6 L3VPN use (equivalent of a per-CE VPN label)
- **End.DX4** Endpoint with decapsulation and IPv4 cross-connect IPv4 L3VPN use (equivalent of a per-CE VPN label)
- **End.DT6** Endpoint with decapsulation and IPv6 table lookup IPv6 L3VPN use (equivalent of a per-VRF VPN label)
- **End.DT4** Endpoint with decapsulation and IPv4 table lookup IPv4 L3VPN use (equivalent of a per-VRF VPN label)
- **End.B6** Endpoint bound to an SRv6 policy SRv6 instantiation of a Binding SID
- **End.B6.Encaps** Endpoint bound to an SRv6 encapsulation Policy SRv6 instantiation of a Binding SID
- **End.BM** Endpoint bound to an SR-MPLS Policy SRv6/SR-MPLS instantiation of a Binding SID
- **End.S** Endpoint in search of a target in table T
VPN (v4 and v6)
& TE
& NFV
Cisco HW with XR and XE
Barefoot HW with P4 code
FD.IO
Linux

Linux

Implementing IPv6 Segment Routing in the Linux Kernel

David Lefran, Olivier Bonaventura
ICTEAM, Université catholique de Louvain
Gand, Belgium
firstname.lastname@ulouvain.be

Abstract
IPv6 Segment Routing is a major IPv6 transition that provides a
solution to the issue of routing that is currently being developed
within the Internet Engineering Task Force (IETF). We present the
implementation of IPv6 Segment Routing in the Linux kernel. We
first describe it in detail and explain how it can be used on host machines
and routers. We then evaluate the performance of implementing IPv6
Segment Routing and the performance of the Linux kernel. The
implementation has been included in the official Linux 3.10 kernel.

1. Introduction
Segment Routing (SR) is a networking technology that provides a
solution to the issue of routing that is currently being developed
within the Internet Engineering Task Force (IETF). We present the
implementation of IPv6 Segment Routing in the Linux kernel. We
first describe it in detail and explain how it can be used on host machines
and routers. We then evaluate the performance of implementing IPv6
Segment Routing and the performance of the Linux kernel. The
implementation has been included in the official Linux 3.10 kernel.

Code Review / wireshark.git / commit
summary | sharing | tag | commit | commitdiff | review
parent: 380a852 | patch
IPv6: Add support for Segment Routing (Type 4) Extension Header

Name | Description | Release
--- | --- | ---
End | Endpoint function | 4.10 (February 2017), srext
End.X | Endpoint function with Layer-3 cross-connect | 4.10 (February 2017), srext
End.T | Endpoint function with specific IPv6 table lookup | 4.14 (November 2017), srext
End.DX2 | Endpoint with decapsulation and Layer-2 cross-connect | 4.14 (November 2017), srext
End.DX6 | Endpoint with decapsulation and IPv6 cross-connect | 4.14 (November 2017), srext
End.DX4 | Endpoint with decapsulation and IPv4 cross-connect | 4.14 (November 2017), srext
End.DT6 | Endpoint with decapsulation and IPv6 table lookup | In development
End.DT4 | Endpoint with decapsulation and IPv4 table lookup | 4.14 (November 2017), srext
End.B6 | Endpoint bound to an SRv6 policy | 4.14 (November 2017), srext
End.BM | Endpoint bound to an SR-MPLS Policy | In development
End.S | Endpoint in search of a target in table T | In development
End.AD | Endpoint to SR-unaware APP via dynamic proxy | srext
End.AM | Endpoint to SR-unaware APP via masquerading | srext

Name | Description | Release
--- | --- | ---
T.Insert | Transit behavior with insertion of an SRv6 Policy | 4.10 (February 2017)
T.Encaps | Transit behavior with encapsulation in an SRv6 policy | 4.10 (February 2017)
T.Encaps.L2 | Transit behavior of the received L2 frame | 4.14 (November 2017)
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>End</td>
<td>Endpoint function</td>
<td>17.04 (April 2017)</td>
</tr>
<tr>
<td>End.X</td>
<td>Endpoint function with Layer-3 cross-connect</td>
<td>17.04 (April 2017)</td>
</tr>
<tr>
<td>End.T</td>
<td>Endpoint function with specific IPv6 table lookup</td>
<td>In development</td>
</tr>
<tr>
<td>End.DX2</td>
<td>Endpoint with decapsulation and Layer-2 cross-connect</td>
<td>17.04 (April 2017)</td>
</tr>
<tr>
<td>End.DX6</td>
<td>Endpoint with decapsulation and IPv6 cross-connect</td>
<td>17.04 (April 2017)</td>
</tr>
<tr>
<td>End.DX4</td>
<td>Endpoint with decapsulation and IPv4 cross-connect</td>
<td>17.04 (April 2017)</td>
</tr>
<tr>
<td>End.DT6</td>
<td>Endpoint with decapsulation and IPv6 table lookup</td>
<td>17.04 (April 2017)</td>
</tr>
<tr>
<td>End.DT4</td>
<td>Endpoint with decapsulation and IPv4 table lookup</td>
<td>17.04 (April 2017)</td>
</tr>
<tr>
<td>End.B6</td>
<td>Endpoint bound to an SRv6 policy</td>
<td>17.04 (April 2017)</td>
</tr>
<tr>
<td>End.BM</td>
<td>Endpoint bound to an SR-MPLS Policy</td>
<td>In development</td>
</tr>
<tr>
<td>End.S</td>
<td>Endpoint in search of a target in table T</td>
<td>In development</td>
</tr>
<tr>
<td>End.AS</td>
<td>Endpoint to SR-unaware APP via static proxy</td>
<td>In development</td>
</tr>
<tr>
<td>End.AM</td>
<td>Endpoint to SR-unaware APP via masquerading</td>
<td>In development</td>
</tr>
<tr>
<td>T.Insert</td>
<td>Transit behavior with insertion of an SRv6 Policy</td>
<td>17.04 (April 2017)</td>
</tr>
<tr>
<td>T.Encaps</td>
<td>Transit behavior with encapsulation in an SRv6 policy</td>
<td>17.04 (April 2017)</td>
</tr>
<tr>
<td>T.Encaps.L2</td>
<td>T.Encaps behavior of the received L2 frame</td>
<td>17.04 (April 2017)</td>
</tr>
</tbody>
</table>
EANTC Interop Showcase

Over 20 companies, including:

- Cisco
- Juniper
- Arista
- Huawei
- Ericsson
- Nokia
- ZTE
- ECI
- NEC

Technologies covered

- SR/SRv6
- EVPN
- PCE
- NC/YANG
- Microwave (IP/MPLS)
- Synchronization

www.eantc.de/en/showcases/mpls_sdn_2018
EANTC SRv6 Interoperability tests

- In this scenario we tested both the END and END.X functions.

- During our test, we verified the expected END and END.X data plane forwarding behavior and IPv6 SR Header (SRH) handling by Cisco and UTStarcom devices for SRv6 encapsulated traffic generated by Spirent TestCenter (STC) and Ixia IxNetwork.

- In our first scenario, UTStarcom’s UAR500 performed the END function and Cisco’s NCS 5500 performed the END.X function. In the second scenario, the roles were inverted accomplishing the same results.

http://www.eantc.de/en/showcases/mpls_sdn_2018
In this lab you will explore how SRv6 works.

In the first part, you will see how BGP extensions allows you to use SRv6 for L3 VPN across the IPv6 network.

In the second part, you will explore the possibilities of the On Demand Nexthop configuration to provide a latency-aware path for L3 VPN traffic.
Stay up to date on Segment Routing

segment-routing.net

amzn.com/B01I58LSUO

ask-segment-routing@cisco.com

linkedin.com/groups/8266623

twitter.com/SegmentRouting

facebook.com/SegmentRouting/
Segment Routing

A source-routing architecture that seeks the right balance between distributed intelligence and centralized optimization. The application steers its packets through an ordered list of instructions and realizes end-to-end policy without creating any per-flow state in the network.

Get Started!  SRv6 video  SR Customer Quotes

Simple  Scalable  Seamless deployment
IETF Docs SRv6

- IPv6 Segment Routing Header (SRH) draft-ietf-6man-segment-routing-header
- IS-IS Extensions to Support Routing over IPv6 Dataplane draft-bashandy-isis-srv6-extensions
- OSPFv3 Extensions for SRv6 draft-li-ospf-ospfv3-srv6-extensions
- BGP Link State extensions for IPv6 Segment Routing(SRv6) draft-dawra-idr-bGPLS-srv6-ext
- Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Data plane (SRv6) draft-ali-spring-srv6-oam
- SRv6 Network Programming draft-filsfils-spring-srv6-network-programming
- SRv6 interoperability report, draft-filsfils-spring-srv6-interop-00