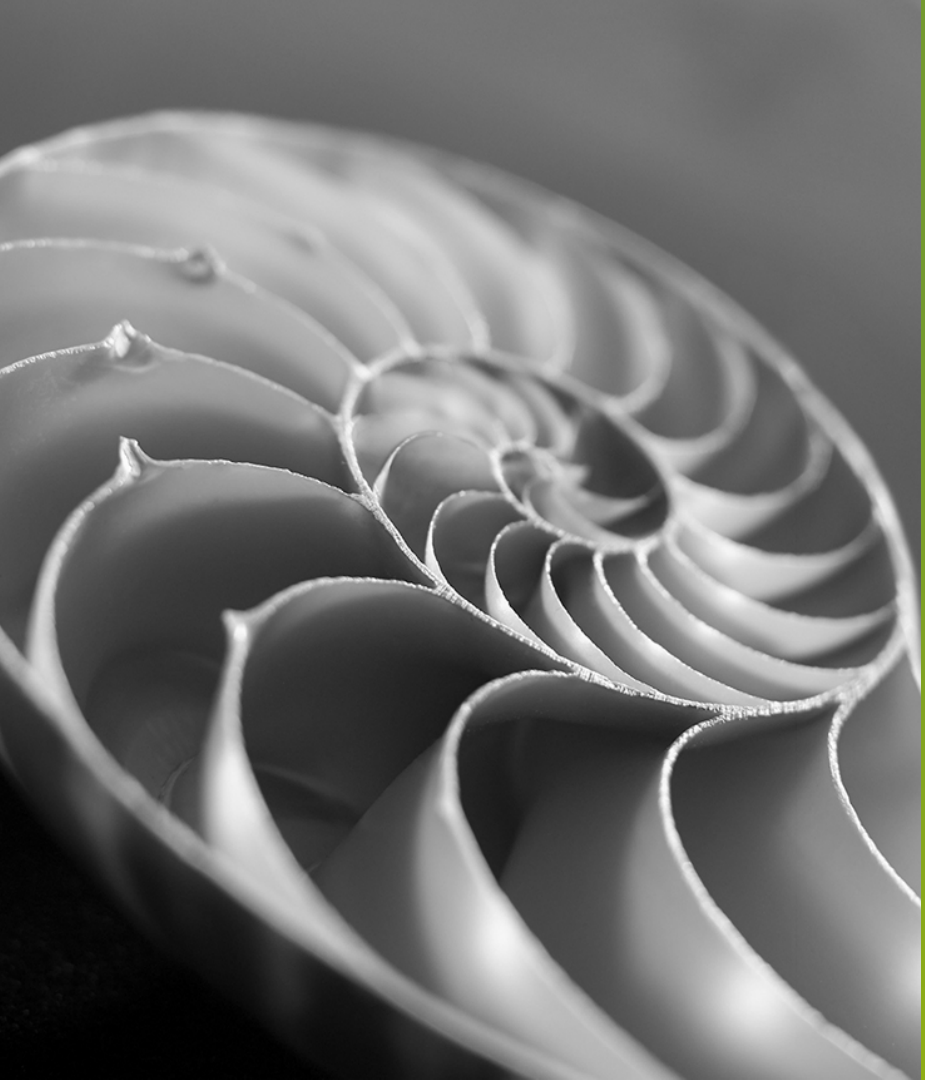
SPRING FOR SERVICE PROVIDER NETWORKS

Aditya Kaul

Professional Services

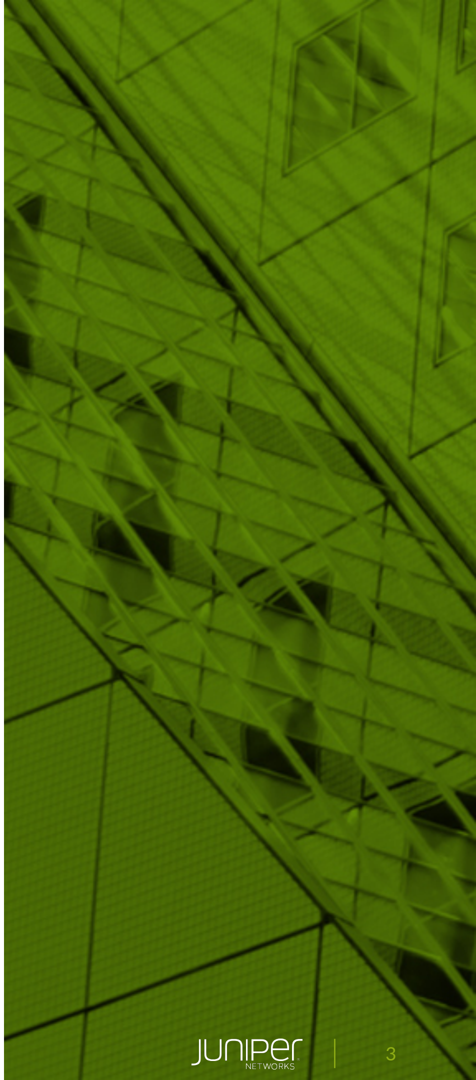
August 2019



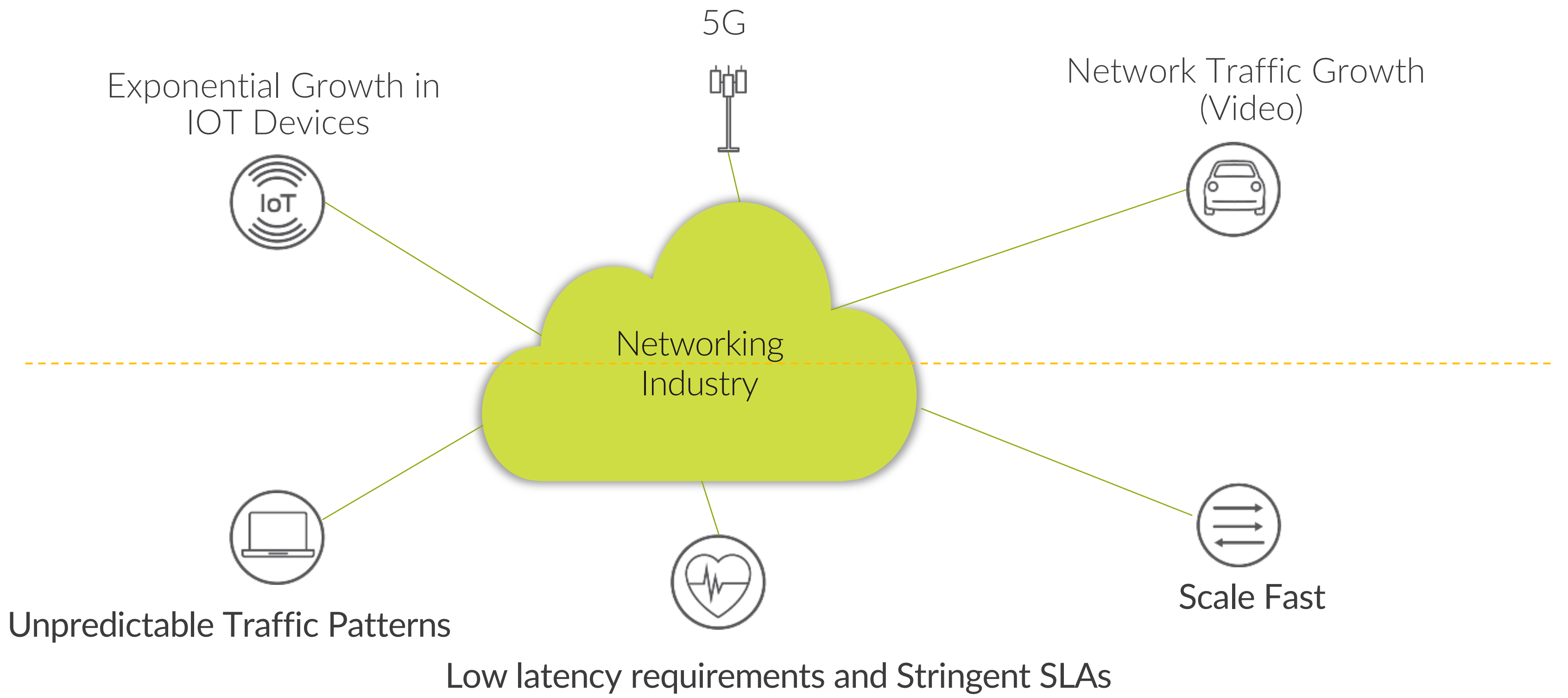
AGENDA

- Introduction
- Spring usecases
- Deployment considerations
- Traffic Engineering
- Flexible Algorithm
- SRv6 and SRv6+
- References

INDUSTRY TRENDS AND DRIVERS



INDUSTRY TRENDS AND CHALLENGES



ENTER THE APPLICATION GRADE NETWORKING ERA

Cloud Grade Routing — **5th Gen Router:** Cloud Grade Networking
- High performance & Agile
- Simple & Scalable
- Highly Available & Responsive

Multiservice Routers — **4th Gen Router:** Integrated Universal Edge

Carrier Grade Routers — **3rd Gen Router:** IP/MPLS routers (performance, reliability)

Extending Networks — **2nd Gen Router:** L3 segmentation on enterprise class elements

Interconnecting Networks — **1st Gen Router:** Specialized software on general purpose compute

CLOUD GRADE ROUTING

Build cloud grade architectures with modern routing technologies

High Performance & Agile

- Maximize network efficiency
- Agile - application driven fabrics across networks
- Customizable stack w/ programmable APIs

Simple & Scalable

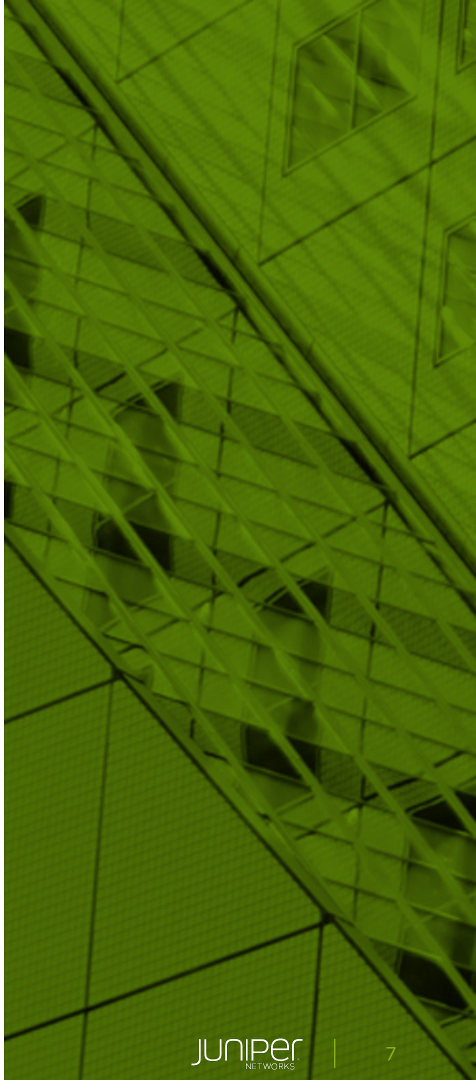
- Simpler architecture
- Scale up or Scale out
- Integration w/ orchestration and controllers

Highly Available & Responsive

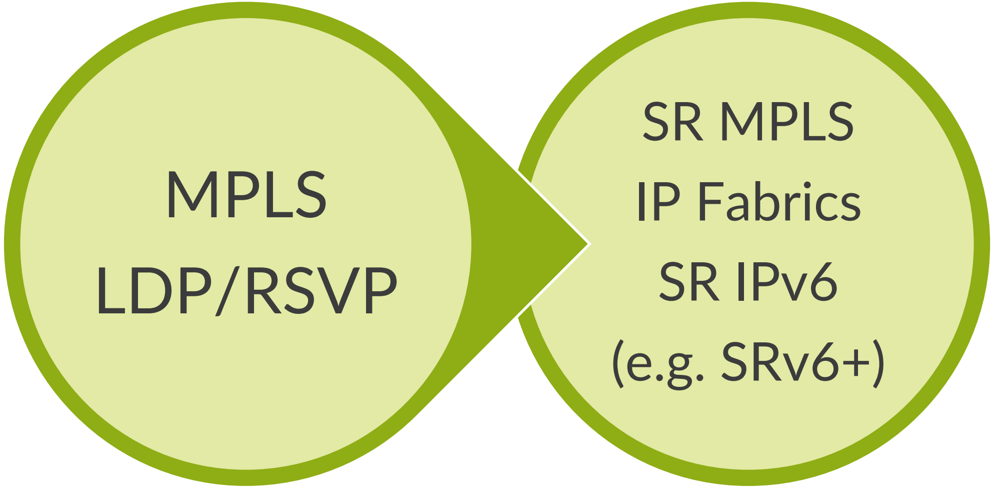
- Built in fault detection, isolation, and recovery
- Rich HA architectures for application SLAs
- Self healing based on real-time telemetry

Transform Network Economics with Modern Architectures

INTRODUCTION

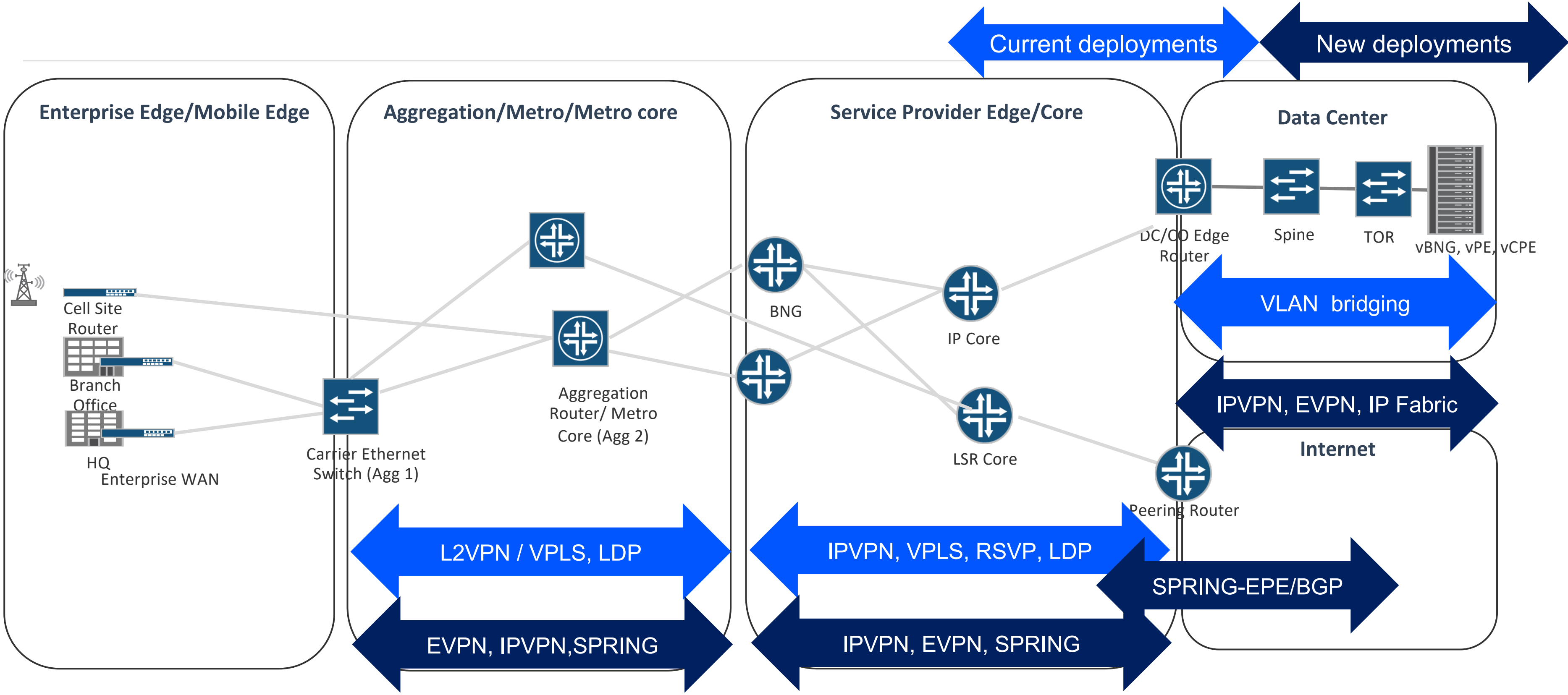


NETWORK TRANSPORT EVOLUTION



Evolution of Network Transport

ROUTING PROTOCOL EVOLUTION



Leverage single *Protocol* across all domains.

INTRODUCTION

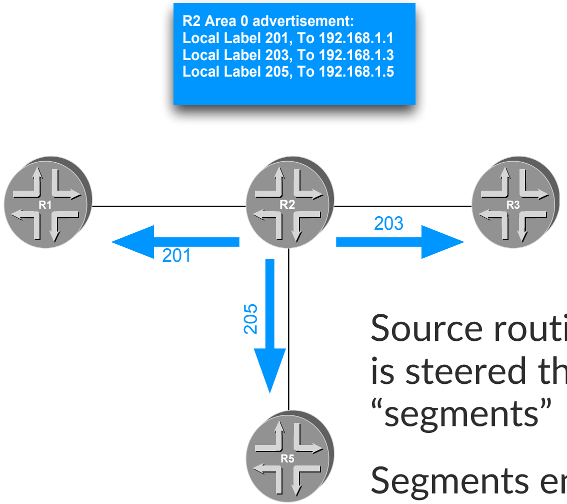
- SPRING a.k.a. Segment Routing
  - Source Packet Routing in Networking
  - Leverages the source routing paradigm; a node steers a packet through an ordered list of instructions, called segments
  - A segment can represent any instruction, topological or service-based
  - Allows to enforce a flow through any topological path and service chain while maintaining per-flow state only at the ingress node
  - A segment is referred to by its Segment Identifier (SID)

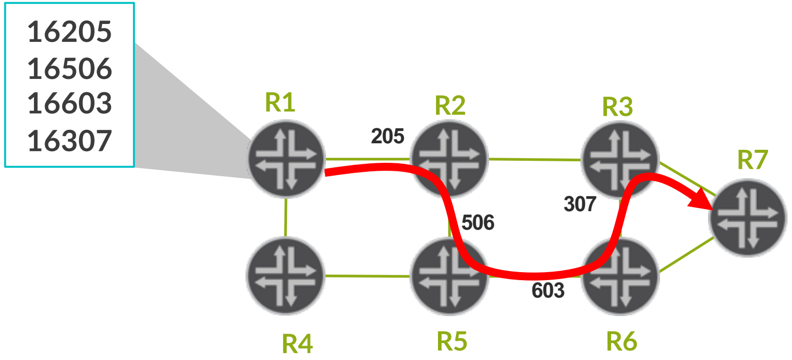
SEGMENT ROUTING (SPRING) PROMISES

- ✓ Simplification and reduction of network signalling components, unify the transport and Service layer
- ✓ Alternative to LDP – labels advertised by the IGP (IS-IS or OSPF), reduced number of protocols in the network
- ✓ Fast restoration (FRR), traffic engineering, and network programmability use cases
- ✓ Alternative to RSVP-TE – more scalable, no path states in the core, no additional protocol for traffic engineering
- ✓ Spring opens new possibilities of network programming via centralised controller and opens new avenues of flexibility, control and feature-rich use cases.

SOURCE PACKET ROUTING IN NETWORKING OVERVIEW

The Two Building Blocks of SPRING





Source routing paradigm where traffic is steered through list of instructions “segments”

Segments encoded as Label or IPv6 address

Ingress Router pushes the series of segments as part of the packet

Accomplishes explicit routing without signaling forwarding state.

1. Advertising Segments in IGP/BGP

2. Forwarding based active segment

CONTROL-PLANES

Distributed	• Segments allocated and signaled by the IGP or BGP • Nodes individually steer packets and compute the source-routed policy
Centralized	• Segments allocated and instantiated by a SR controller • SR controller decides to steer packets, computes the source-routed policies, and programs the network via NETCONF, PCEP, or BGP
Hybrid	• Combines a base distributed control-plane with a centralized controller

SEGMENT TYPES

- Link-State IGP Segments
  - Represent IGP prefixes or IGP adjacencies
  - Signaled by IS-IS or OSPF
- BGP Segments
  - Correspond to BGP peers
  - Signaled by BGP
- Binding Segments
  - Refer to SR policies

IGP SEGMENT TYPES

- Adjacency Segments
  - IGP Segment attached to an unidirectional adjacency or set of adjacencies
  - One-hop path to immediate neighbor
  - Local – only the originating router assigns a label to the local segment
- Prefix Segments
  - IGP segment attached to an IGP prefix
  - Identifies shortest-path computed by the IGP to the related prefix
  - Multi-hop path tunnels to all other nodes
  - Global – every router in the domain assigns a (local) label to the (global) segment
- Node Segments
  - IGP prefix segment which identifies a specific router (e.g. a loopback)
  - Enforces shortest-path forwarding to the related node
- Anycast Segments
  - IGP prefix segment which identifies a set of routers
  - Enforces shortest-path forwarding towards the closest node of the anycast set

BGP PREFIX-SID

Distribute Service PE loopback with prefix-SID in BGP-LU – Seamless MPLS architecture

Why BGP prefix-SID?

- Prefix to label index from SRGB
- If SRGB is same across domains, Prefix to label binding is same
- BGP prefix-SID is an optional transitive attribute and normal BGP-LU label is used for next-hop programming
- Forwarding should work if device is not the edge device originating BGP prefix-SID

SPRING ENCAPSULATION TYPES

MPLS

- SR header is an MPLS label stack
- Each label in the stack represents a segment

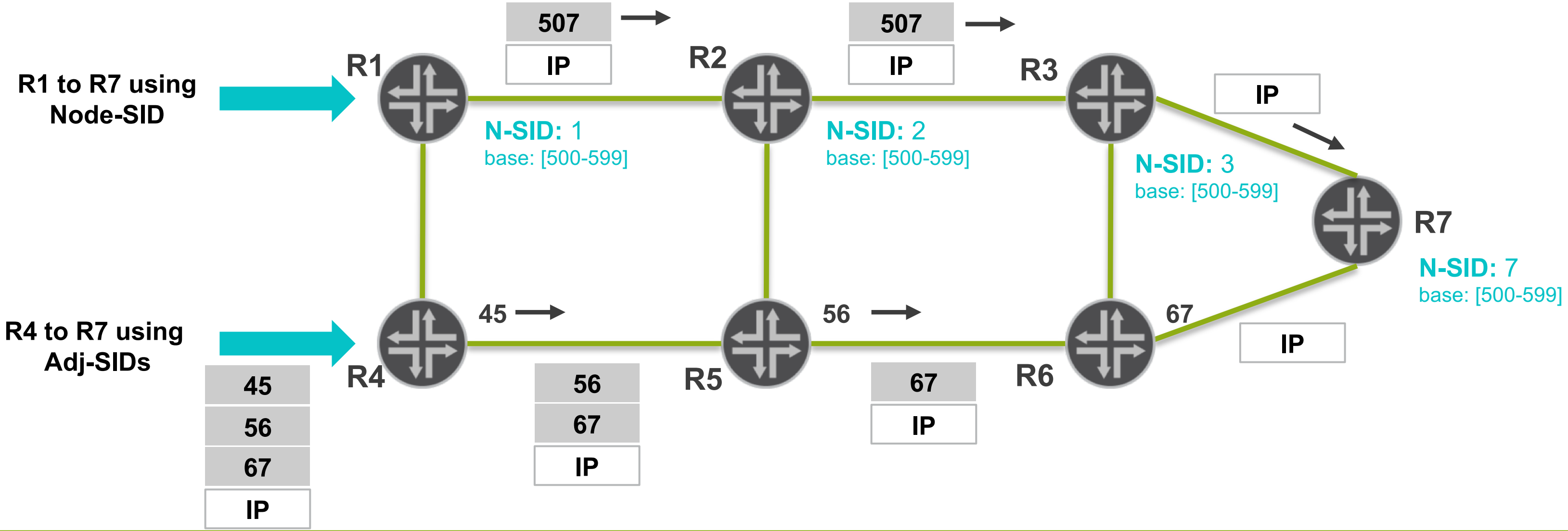
IPv6

- SR Header is an IPv6 header with a Segment Routing Extension Header or Compressed Routing Header
- SRH and CRH contains a list of IPv6 addresses
- Each IPv6 address represents a segment

BASIC FORWARDING EXAMPLES

Prefix/Node-SID forwarding (using SRGB)

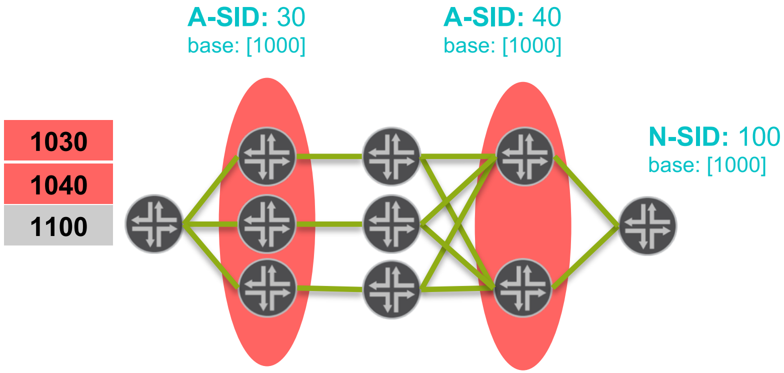
- R1 shortest path to R7 is via R2.
- R2 expects a label value equal to {R2 label-base + index of destination}
  R1 => R2 label = 507 {500 + 7}



ANYCAST/BINDING SIDS

Anycast-SIDs

- have domain-wide significance
- define a set of nodes via a non-uniquely announced prefix
- forwarding choice is made via IGP SPF
- can use ECMP for forwarding
- add redundancy, enable load balancing
- commonly represent a set of geographically close nodes (e.g.: metro)

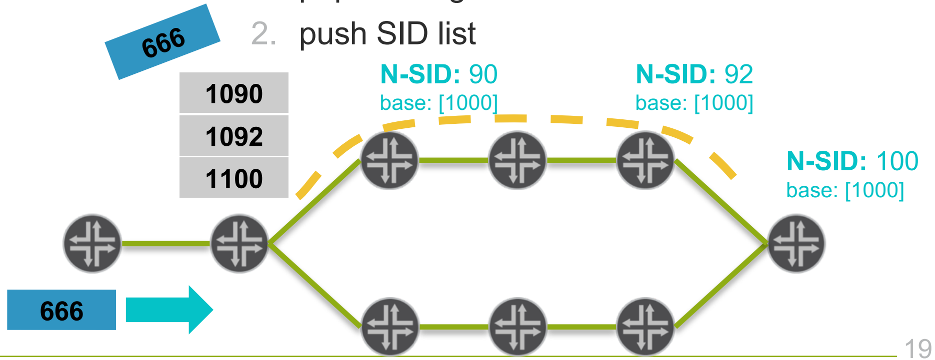


Binding-SIDs

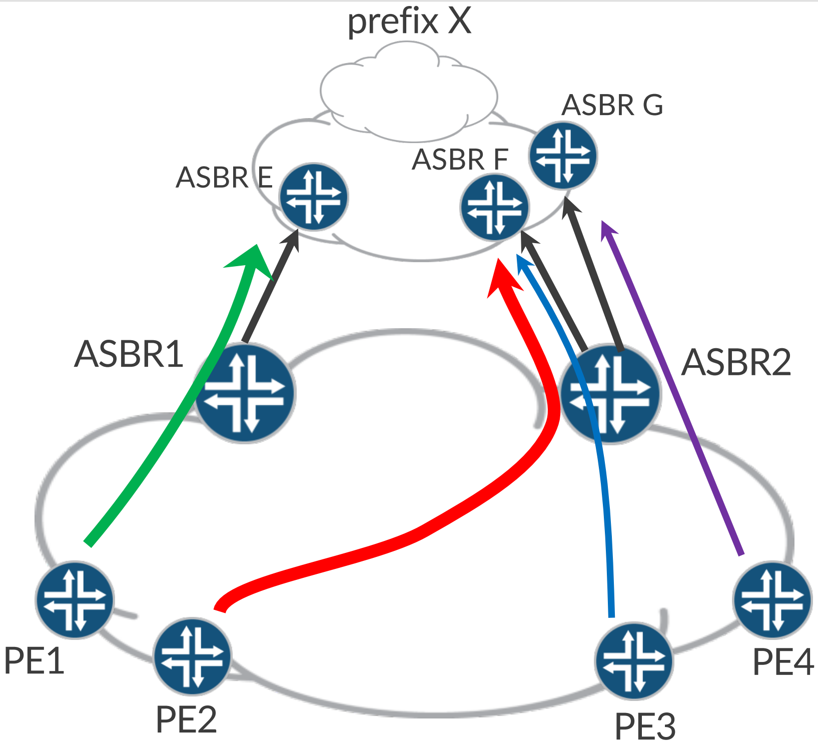
- have node-local significance
- are bound to other SR paths
- enable an SR path to include another SR path by reference
- are useful for scaling the SID stack at ingress

Binding-SID forwarding operation:

1. pop Binding-SID label
2. push SID list



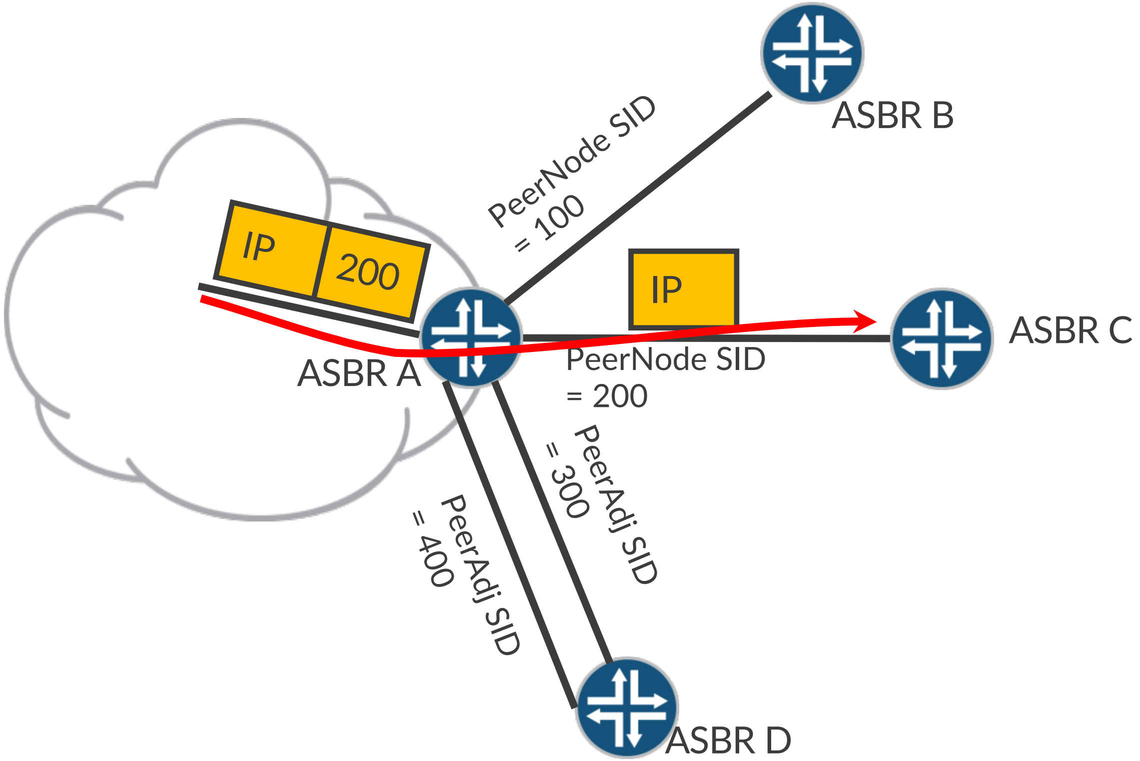
EGRESS PEERING ENGINEERING (EPE)



Peering link on ASBR1 is approaching congestion outbound.

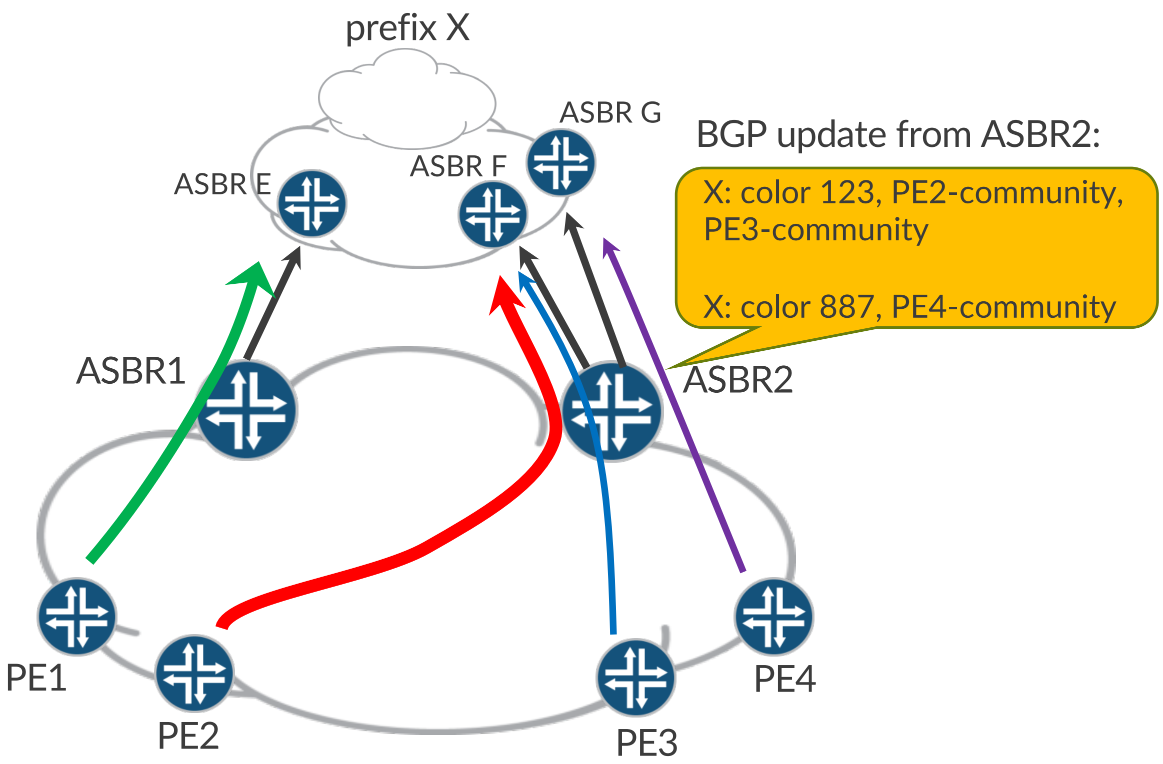
NorthStar moves traffic from PE2 to prefix X onto one of the peering links on ASBR2.

EPE INGREDIENTS: PEER-SID



- SID values manually configured on ASBR A
- Advertised via BGP-LS to NorthStar
- SID label steers packet towards desired peering link
- ASBR A pops label, so plain IP packet is sent on the peering link

EGRESS PEERING ENGINEERING (EPE)



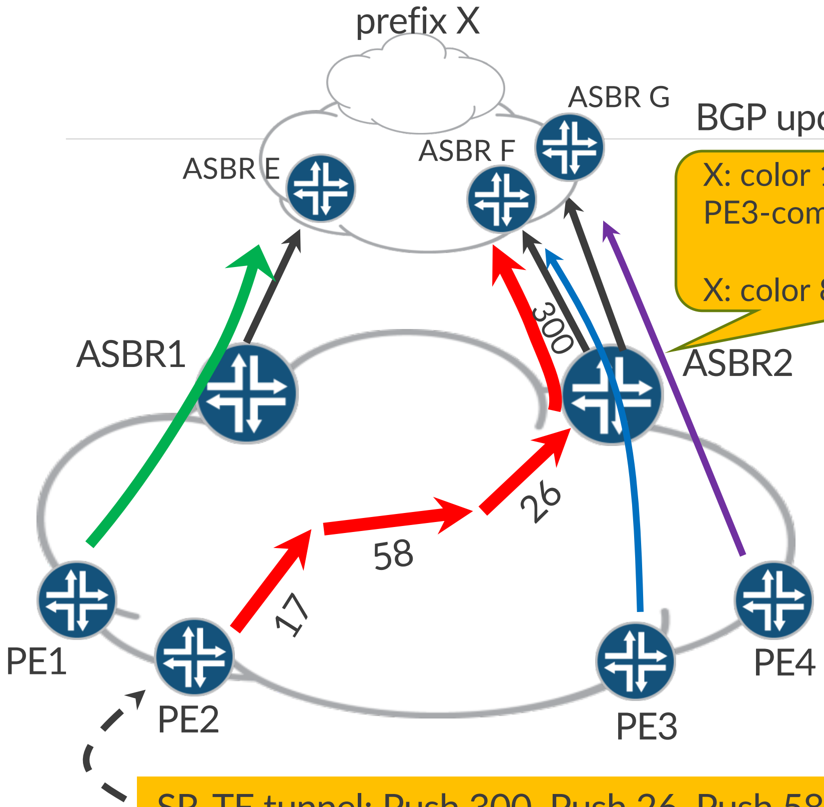
NorthStar uses Programmable RPD (pRPD) to manipulate attributes of prefix X on the ASBRs:

A community, specific to each PE, is added to the desired version of the path to prefix X. Also a color community is added.

BGP Add-Path is used, so that all paths corresponding to prefix X are propagated.

Each PE has a BGP import policy to prefer the version of the path that has its own PE-specific community attached.

EGRESS PEERING ENGINEERING (EPE)



BGP update from ASBR2:

X: color 123, PE2-community, PE3-community

X: color 887, PE4-community

SR-TE tunnel: Push 300, Push 26, Push 58, Push 17 (top),
NH=ASBR2,
Color community: 123

NorthStar installs SR-TE tunnels on each ingress PE.

Bottom label in stack is the PeerAdjSID or PeerNodeSID needed to steer the packet onto the required egress link.

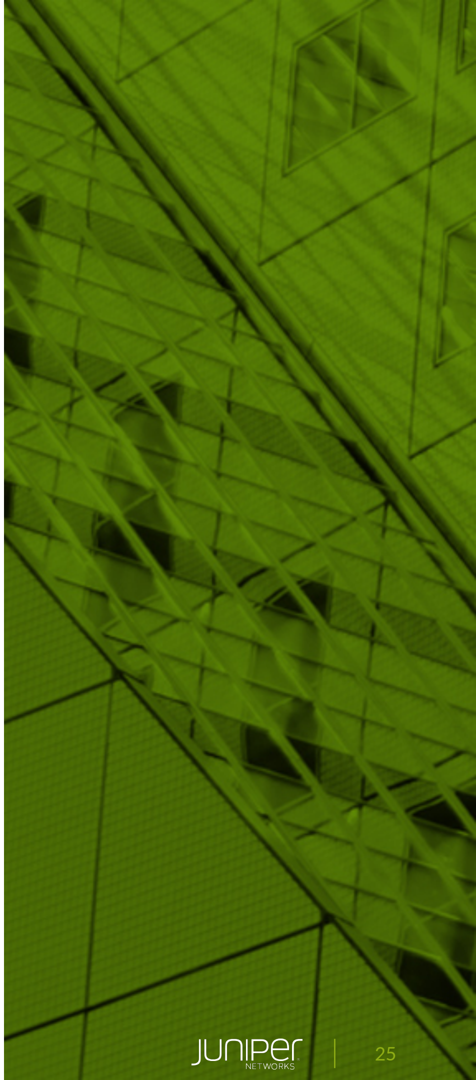
If traffic to multiple prefixes are required to use the same peering link, it can use the same tunnel.

Regardless of whether ASBR2 is doing next-hop self or not, the bottom label in the stack steers the packet onto the desired egress link.

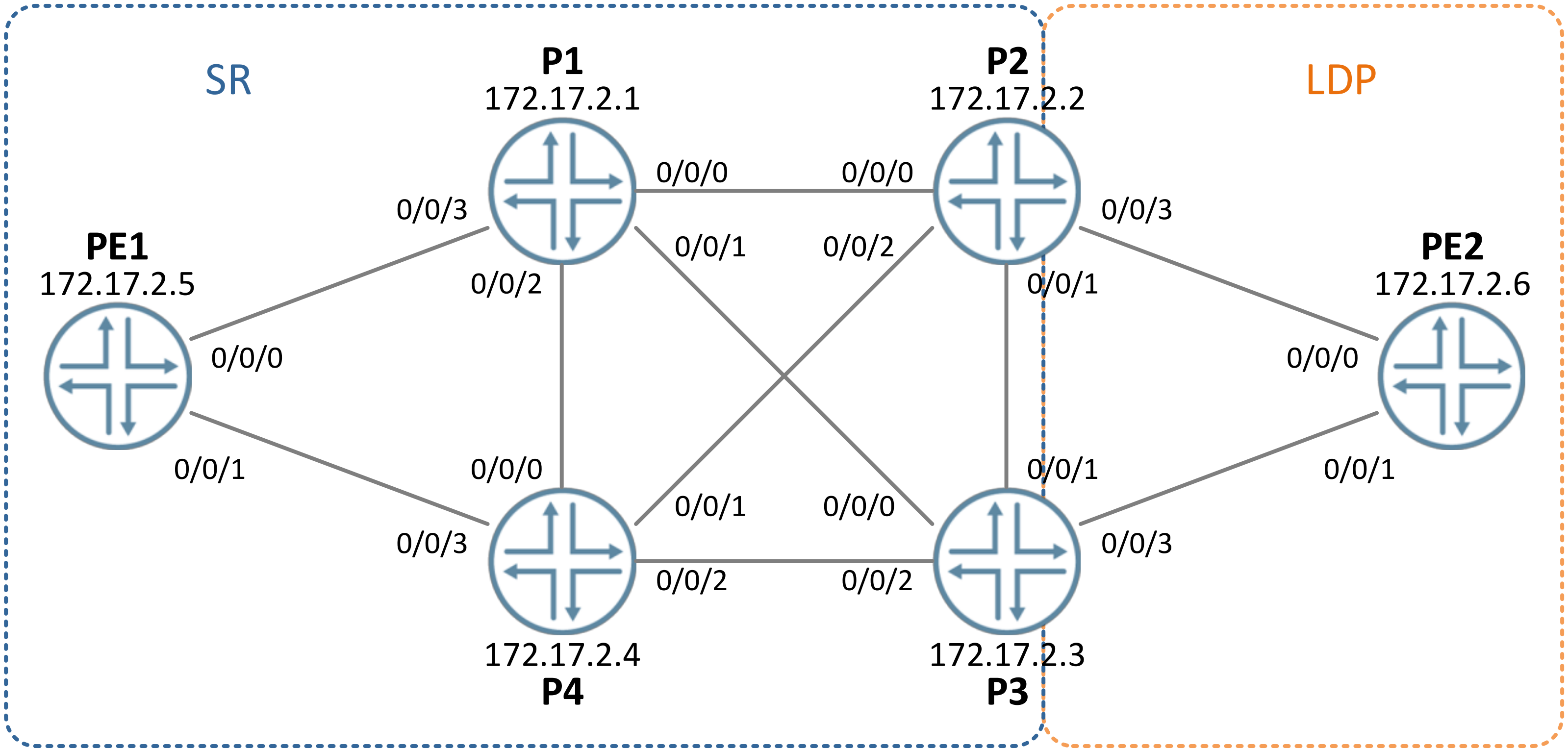
DIFFERENT FLAVORS FOR SPRING

	SR-MPLS	SRv6	SR-MPLS over UDP/IP
IGP	IPv4, IPv6	IPv6	IPv4, IPv6
Segment Identifier	Label (20 bits)	IPv6 address (128 bits)	Label (20 bits)
Forwarding	Label Switching	IPv6, SRv6	IPv4, IPv6
Forwarding operation	Push, Pop, Swap	IPv6 header update	IP (v4/v6) over UDP
SR TE Path	Stack of Labels (20 bit SIDs)	Stack of IPv6 addresses	Stack of Labels
Entropy	Entropy Label	Flow Label	UDP source Port
Integration into existing network	All nodes need SR	Seamless integration	Seamless integration
Draft	draft-ietf-spring-segment-routing-mpls	draft-ietf-6man-segment-routing-header	draft-ietf-mpls-sr-over-ip

SPRING CONFIGURATION AND VERIFICATION – IS-IS EXAMPLE



SPRING LAB SETUP – VMX / JUNOS 18.1R1



IS-IS CONFIGURATION

```
protocols {
    isis {
        backup-spf-options {                          ← LFA configuration options
            use-post-convergence-lfa;                 ← enables TI-LFA
            use-source-packet-routing;                ← enables use of node segments for LFA
        }
        source-packet-routing {                       ← enables SPRING
            srgb start-label 800000 index-range 5000; ← configurable SRGB
            node-segment {                            ← enables node segments
                ipv4-index 1001;                      ← IPv4 node segment index
                ipv6-index 2001;                      ← IPv6 node segment index
            }
        }
        level 1 disable;
        level 2 wide-metrics-only;
        interface all {
            point-to-point;
            level 2 {
                post-convergence-lfa;                 ← enables interface for TI-LFA
            }
        }
    }
```

IS-IS VERIFICATION

```
root@vmxdockerlight_p1_1> show isis overview
Instance: master
  <...>
  IPv4 is enabled, IPv6 is enabled, SPRING based MPLS is enabled
  Traffic engineering: enabled
  <...>
  Source Packet Routing (SPRING): Enabled
    SRGB Config Range:
      SRGB Start-Label : 800000, SRGB Index-Range : 5000
    SRGB Block Allocation: Success
      SRGB Start Index : 800000, SRGB Size : 5000, Label-Range: [ 800000, 804999 ]
    Node Segments: Enabled
      Ipv4 Index : 1001, Ipv6 Index : 2001
  Post Convergence Backup: Enabled
    Max labels: 3, Max spf: 100, Max Ecmp Backup: 1
  Level 1
    <...>
    Source Packet Routing is enabled
  Level 2
    <...>
    Source Packet Routing is enabled
```

LABEL ALLOCATION

```
root@vmxdockerlight_p1_1> show mpls label usage

Label space Total    Available        Applications
LSI         994984   994975 (100.00%) BGP/LDP VPLS with no-tunnel-services, BGP L3VPN with
                                      vrf-table-label
Block       994984   994975 (100.00%) BGP/LDP VPLS with tunnel-services, BGP L2VPN
Dynamic     994984   994975 (100.00%) RSVP, LDP, PW, L3VPN, RSVP-P2MP, LDP-P2MP, MVPN, EVPN, BGP
Static      48576    48576  (100.00%) Static LSP, Static PW
Effective Ranges
Range name  Shared with Start   End
Dynamic     16      799999
Dynamic     805000  999999
Static      1000000 1048575
SRGB        800000  804999
Configured Ranges
Range name  Shared with Start   End
Dynamic     16      799999
Dynamic     805000  999999
Static      1000000 1048575
SRGB        800000  804999
```

IS-IS ROUTER CAPABILITIES SR SUB-TLV

```
root@vmxdockerlight_p1_1> show isis database vmxdockerlight_p1_1.00-00 extensive
IS-IS level 2 link-state database:

vmxdockerlight_p1_1.00-00 Sequence: 0x1b8, Checksum: 0x952f, Lifetime: 783 secs
  <...>
  TLVs:
    <...>
    Router Capability:  Router ID 172.17.2.1, Flags: 0x00
      SPRING Capability - Flags: 0xc0(I:1,V:1), Range: 5000, SID-Label: 800000
      SPRING Algorithm - Algo: 0
```

I-Flag: MPLS IPv4 flag – if set, the router is capable of processing SR MPLS encapsulated IPv4 packets on all interfaces
V-Flag: MPLS IPv6 flag - if set, the router is capable of processing SR MPLS encapsulated IPv6 packets on all interfaces
Range: number of SRGB elements
SID-Label: first value of the SRGB
Algo 0 = SPF

ADJACENCY SEGMENTS

```
root@vmxdockerlight_p1_1> show isis adjacency detail
vmxdockerlight_p2_1
  Interface: ge-0/0/0.0, Level: 2, State: Up, Expires in 21 secs
  <...>
  Restart capable: Yes, Adjacency advertisement: Advertise
  IP addresses: 172.22.64.2
  Level 2 IPv4 Adj-SID: 19

vmxdockerlight_p3_1
  Interface: ge-0/0/1.0, Level: 2, State: Up, Expires in 25 secs
  <...>
  Restart capable: Yes, Adjacency advertisement: Advertise
  IP addresses: 172.22.72.3
  Level 2 IPv4 Adj-SID: 18

root@vmxdockerlight_p1_1> show route table mpls.0
18                 *[L-ISIS/14] 3d 19:35:55, metric 0
                    > to 172.22.72.3 via ge-0/0/1.0, Pop
                      to 172.22.65.3 via ge-0/0/2.0, Swap 801003

19                 *[L-ISIS/14] 3d 19:35:55, metric 0
                    > to 172.22.64.2 via ge-0/0/0.0, Pop
                      to 172.22.65.3 via ge-0/0/2.0, Swap 801002
```

IS-IS ADJ-SID SUB-TLV

```
root@vmxdockerlight_p1_1> show isis database vmxdockerlight_p1_1.00-00 extensive
IS-IS level 2 link-state database:

vmxdockerlight_p1_1.00-00 Sequence: 0x1b8, Checksum: 0x952f, Lifetime: 783 secs
  <...>
  TLVs:
    <...>
    Extended IS Reachability TLV, Type: 22, Length: 160
    IS extended neighbor: vmxdockerlight_p2_1.00, Metric: default 10 SubTLV len: 29
      <...>
      P2P IPV4 Adj-SID - Flags:0x70(F:0,B:1,V:1,L:1,S:0,P:0), Weight:0, Label: 19
      P2P IPv4 Adj-SID:      19, Weight:   0, Flags: -BVL--
    IS extended neighbor: vmxdockerlight_p3_1.00, Metric: default 10 SubTLV len: 29
      <...>
      P2P IPV4 Adj-SID - Flags:0x70(F:0,B:1,V:1,L:1,S:0,P:0), Weight:0, Label: 18
      P2P IPv4 Adj-SID:      18, Weight:   0, Flags: -BVL–
```

F-Flag: Address-Family flag – outgoing encapsulation, 0 = IPv4, 1 = IPv6
B-Flag: Backup flag – if set, Adj-SID is eligible for protection
V-Flag: Value flag – if set, Adj-SID carries a value (instead of an index)
L-Flag: Local Flag – if set, value/index has local significance.
S-Flag: Set flag – if set, Adj-SID refers to a set of adjacencies
P-Flag: Persistent flag – if set, Adj-SID is persistently allocated, i.e. remains across router restart and/or interface flap

IS-IS ADVERTISEMENTS VERIFICATION

```
May 14 08:03:00.825040 Received L2 LSP vmxdockerlight_p1_1.00-00, on interface ge-0/0/0.0
May 14 08:03:00.825118     <...>
May 14 08:03:00.825262     IS neighbor vmxdockerlight_p2_1.00, metric: 10, TLV22 tlvlen: 160, subtlv: 29
May 14 08:03:00.825282       IP address: 172.22.64.3
May 14 08:03:00.825292       Neighbor IP address: 172.22.64.2
May 14 08:03:00.825300       interface indices: 333, 333
May 14 08:03:00.825308       IP address: 0.0.1.77
May 14 08:03:00.825317       P2P Adj-SID (Len: 5, Flags:0x70, Weight:0, Label:19)
May 14 08:03:00.825326     IS neighbor vmxdockerlight_p3_1.00, metric: 10, TLV22 tlvlen: 120, subtlv: 29
May 14 08:03:00.825335       IP address: 172.22.72.2
May 14 08:03:00.825343       Neighbor IP address: 172.22.72.3
May 14 08:03:00.825349       interface indices: 334, 333
May 14 08:03:00.825357       IP address: 0.0.1.78
May 14 08:03:00.825364       P2P Adj-SID (Len: 5, Flags:0x70, Weight:0, Label:18)
May 14 08:03:00.825372     <...>
May 14 08:03:00.825778     IP unicast prefix: 172.17.2.1/32 metric 10 up
May 14 08:03:00.825944       8 bytes of subtlvs
May 14 08:03:00.825988       Node SID, Flags:0x40, Algo:SPF(0), Value:1001
May 14 08:03:00.826008 L2 LSP id vmxdockerlight_p1_1.00-00, 19 bytes of Router-Capability TLV received
May 14 08:03:00.826024       Spring Capabilities - Len:9, Flags:0xc0, Range:5000, Start-Label:800000
May 14 08:03:00.826031       Spring Algorithms - Algo:0
```

STATIC ADJACENCY SEGMENTS

- By default, Adjacency SIDs are allocated dynamically
- Static Adjacency SID configuration option

```
protocols {
    isis {
        <...>
        interface ge-0/0/1.0 {
            point-to-point;
            level 2 {
                post-convergence-lfa;
                ipv4-adjacency-segment {
                    unprotected label 1000302;      ← label from static label pool, or from SRGB with index
                }
            }
```

```
root@vmxdockerlight_p3_1> show isis adjacency detail
vmxdockerlight_p2_1
  Interface: ge-0/0/1.0, Level: 2, State: Up, Expires in 22 secs
  <...>
  Level 2 IPv4 unprotected Adj-SID: 1000302, Flags: --VL-P      ← Persistent (P) flag set
```

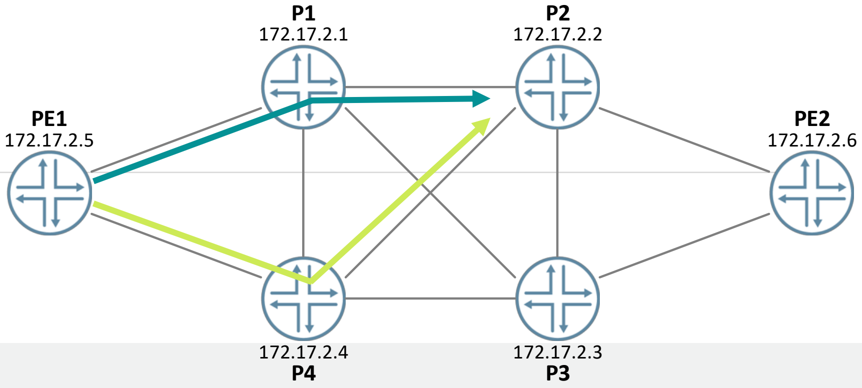
NODE SEGMENTS & IS-IS PREFIX-SID SUB-TLVS

```
root@vmxdockerlight_p1_1> show isis database vmxdockerlight_p1_1.00-00 extensive
IS-IS level 2 link-state database:

vmxdockerlight_p1_1.00-00 Sequence: 0x1b8, Checksum: 0x952f, Lifetime: 783 secs
  <...>
  TLVs:
    <...>
    IP extended prefix: 172.17.2.1/32 metric 10 up
      8 bytes of subtlvs
      Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 1001
```

R-Flag: Re-advertisement flag – if set, prefix has been propagated from leaking or redistribution
N-Flag: Node-SID flag – if set, Prefix-SID is a Node-SID
P-Flag: no-PHP flag – if set, penultimate hop must not pop the Prefix-SID
E-Flag: Explicit-Null Flag – if set, upstream neighbor must replace the Prefix-SID with an Explicit-NULL value
V-Flag: Value flag – if set, Prefix-SID carries a value (instead of an index)
L-Flag: Local Flag – if set, value/index carried by the Prefix-SID has local significance

NODE SEGMENTS EXAMPLE

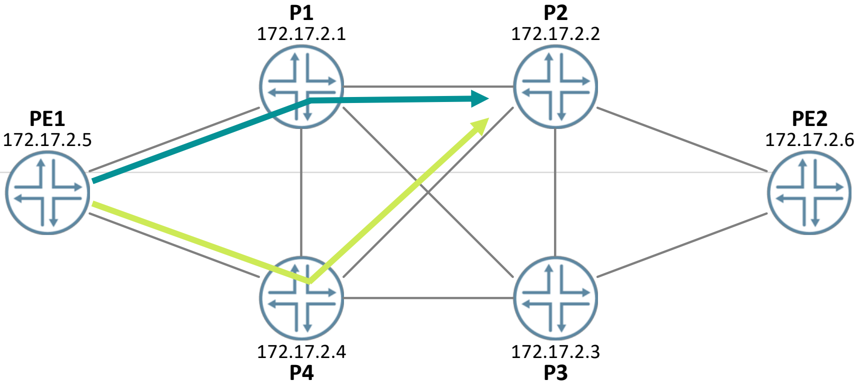
- SR path from PE1 to P2
- Label push at ingress PE1




```
root@vmxdockerlight_pe1_1> show isis database extensive

vmxdockerlight_p1_1.00-00 Sequence: 0x1c5, Checksum: 0x7b3c, Lifetime: 880 secs
    SPRING Capability - Flags: 0xc0(I:1,V:1), Range: 5000, SID-Label: 800000        ← P1 SRGB first label

vmxdockerlight_p4_1.00-00 Sequence: 0x1ca, Checksum: 0xe3a4, Lifetime: 742 secs
    SPRING Capability - Flags: 0xc0(I:1,V:1), Range: 5000, SID-Label: 800000        ← P4 SRGB first label

vmxdockerlight_p2_1.00-00 Sequence: 0x1cb, Checksum: 0x4a59, Lifetime: 427 secs
    IP extended prefix: 172.17.2.2/32 metric 10 up                     ← P2 loopback
      Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 1002  ← P2 Node SID

root@vmxdockerlight_pe1_1> show route 172.17.2.2 table inet.3

172.17.2.2/32       *[L-ISIS/14] 2d 02:41:12, metric 30
                       to 172.22.66.3 via ge-0/0/0.0, Push 801002← transmit label to P1 = P1 SRGB first label + P2 Node SID
                     > to 172.22.67.3 via ge-0/0/1.0, Push 801002← transmit label to P4 = P4 SRGB first label + P2 Node SID
```

NODE SEGMENTS EXAMPLE

- Label pop/swap operation at intermediate nodes P1 and P4




```
root@vmxdockerlight_p1_1> show route table mpls.0 label 801002

801002             *[L-ISIS/14] 00:40:06, metric 20
                    > to 172.22.64.2 via ge-0/0/0.0, Pop              ← direct link to P2: pop transport label
                      to 172.22.65.3 via ge-0/0/2.0, Swap 801002      ← backup link via P4: swap transport label

root@vmxdockerlight_p4_1> show route table mpls.0 label 801002

801002             *[L-ISIS/14] 00:00:04, metric 20
                    > to 172.22.73.2 via ge-0/0/1.0, Pop              ← direct link to P2: pop transport label
                      to 172.22.65.2 via ge-0/0/0.0, Swap 801002      ← backup link via P1: swap transport label
```

PREFIX AND ANYCAST SEGMENTS: CONFIGURATION

- Configured via routing policy
- For an Anycast SID, configure the same Prefix SID on multiple routers

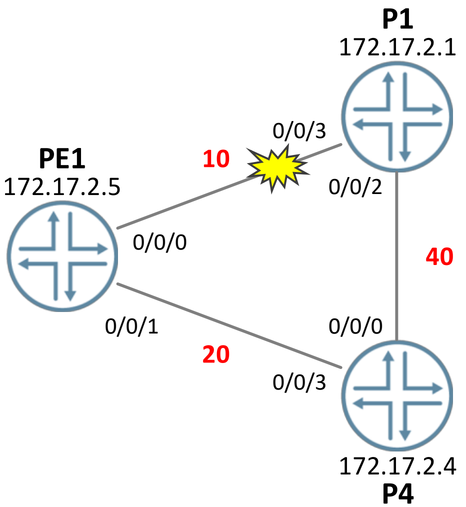
```
protocols {
    isis {
        export PREFIX-SIDS;
        <...>
    }
}
policy-options {
    policy-statement PREFIX-SIDS {
        term 1 {
            from {
                route-filter 172.20.1.1/32 exact;
            }
            then {
                prefix-segment index 2011;
                accept;
            }
        }
    }
```

PREFIX AND ANYCAST SEGMENTS: VERIFICATION

```
root@vmxdockerlight_p2_1> show isis database vmxdockerlight_p2_1.00-00 extensive

vmxdockerlight_p2_1.00-00 Sequence: 0x77b, Checksum: 0x617, Lifetime: 615 secs
  <...>
  TLVs:
    <...>
    IP extended prefix: 172.20.1.1/32 metric 0 up
      8 bytes of subtlvs
      Prefix SID, Flags: 0x00(R:0,N:0,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 2011


root@vmxdockerlight_p1_1> show route table mpls.0

mpls.0: 31 destinations, 31 routes (31 active, 0 holddown, 0 hidden)

802011             *[L-ISIS/14] 00:16:47, metric 10
                    > to 172.22.64.2 via ge-0/0/0.0, Pop
                      to 172.22.72.3 via ge-0/0/1.0, Swap 801002
```

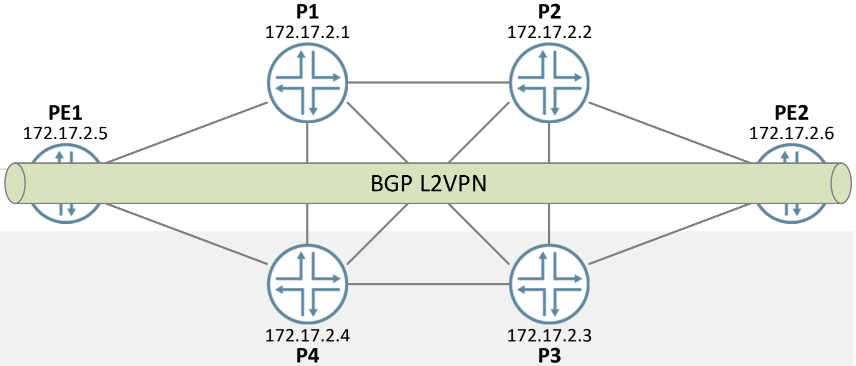
TOPOLOGY-INDEPENDENT LFA EXAMPLE

- Without TI-LFA, there is no LFA from PE1 to P1 due to the higher metrics
- With TI-LFA, PE1 creates a backup path via P4 using the Adjacency SID label for the link from P4 to P1




```
root@vmxdockerlight_pe1_1> show route 172.17.2.1 detail
inet.0: 29 destinations, 29 routes (29 active, 0 holddown, 0 hidden)

172.17.2.1/32 (1 entry, 1 announced)
        *IS-IS  Preference: 18
                <...>
                Next hop: 172.22.66.3 via ge-0/0/0.0 weight 0x1, selected
                <...>
                Next hop: 172.22.67.3 via ge-0/0/1.0 weight 0xf000
                Label operation: Push 68

root@vmxdockerlight_p4_1> show route table mpls.0 label 68

68                 *[L-ISIS/14] 1d 00:55:47, metric 0
                    > to 172.22.65.2 via ge-0/0/0.0, Pop
                      to 172.22.67.2 via ge-0/0/3.0, Swap 801001
```

VPN SERVICES VIA SPRING: BGP L2VPN EXAMPLE




```
root@vmxdockerlight_pe1_1> show l2vpn connections
Layer-2 VPN connections:
Instance: l2vpn-test
  Local site: PE-005 (5)
    connection-site           Type  St     Time last up          # Up
    6                         rmt   Up     May 14 10:24:39 2018
      Remote PE: 172.17.2.6, Negotiated control-word: Yes (Null)
      Incoming label: 25, Outgoing label: 22
      Local interface: ge-0/0/2.0, Status: Up, Encapsulation: ETHERNET
      Flow Label Transmit: No, Flow Label Receive: No

root@vmxdockerlight_pe2_1> show l2vpn connections
Layer-2 VPN connections:
Instance: l2vpn-test
  Local site: PE-006 (6)
    connection-site           Type  St     Time last up          # Up
    5                         rmt   Up     May 14 10:49:12 2018
      Remote PE: 172.17.2.5, Negotiated control-word: Yes (Null)
      Incoming label: 22, Outgoing label: 25
      Local interface: ge-0/0/2.0, Status: Up, Encapsulation: ETHERNET
      Flow Label Transmit: No, Flow Label Receive: No
```

VPN SERVICES VIA SPRING: BGP L2VPN EXAMPLE

```
root@vmxdockerlight_pe1_1> show route table l2vpn-test.l2vpn.0 protocol bgp detail

l2vpn-test.l2vpn.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
 3320:6:6:5/96 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                <...>
                Protocol next hop: 172.17.2.6
                <...>
                Label-base: 22, range: 2, status-vector: 0x0, offset: 5

root@vmxdockerlight_pe1_1> show route 172.17.2.6 table inet.3

172.17.2.6/32      *[L-ISIS/14] 02:04:07, metric 40
                      to 172.22.66.3 via ge-0/0/0.0, Push 800006
                    > to 172.22.67.3 via ge-0/0/1.0, Push 800006

root@vmxdockerlight_pe1_1> show route table mpls.0 protocol l2vpn

25                 *[L2VPN/7] 02:05:25
                    > via ge-0/0/2.0, Pop       Offset: 4
ge-0/0/2.0         *[L2VPN/7] 02:05:25, metric2 40
                    > to 172.22.66.3 via ge-0/0/0.0, Push 22, Push 800006(top) Offset: 252
                      to 172.22.67.3 via ge-0/0/1.0, Push 22, Push 800006(top) Offset: 252
```

← L2VPN route (format = RD:Site ID:offset)

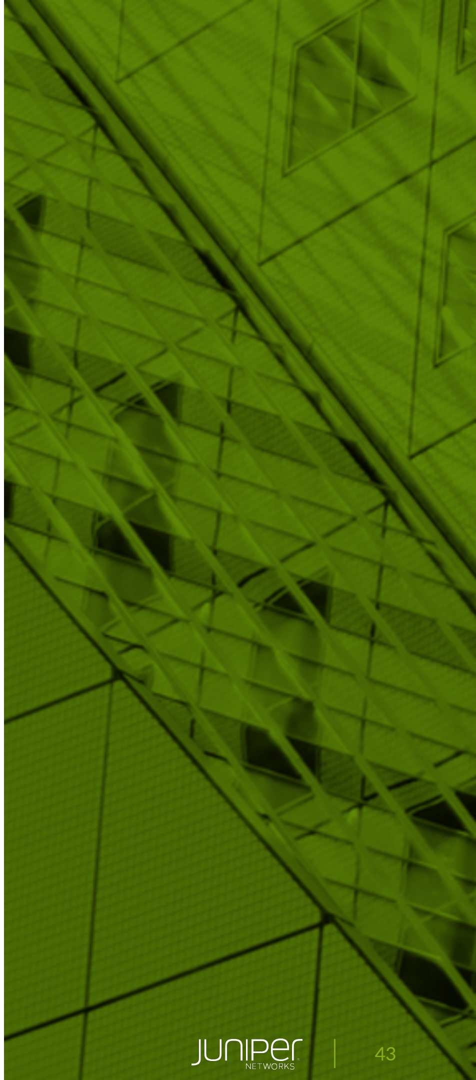
← BGP next-hop to egress PE resolved via inet.3

← outgoing L2VPN label = remote label base + local site ID – remote label offset

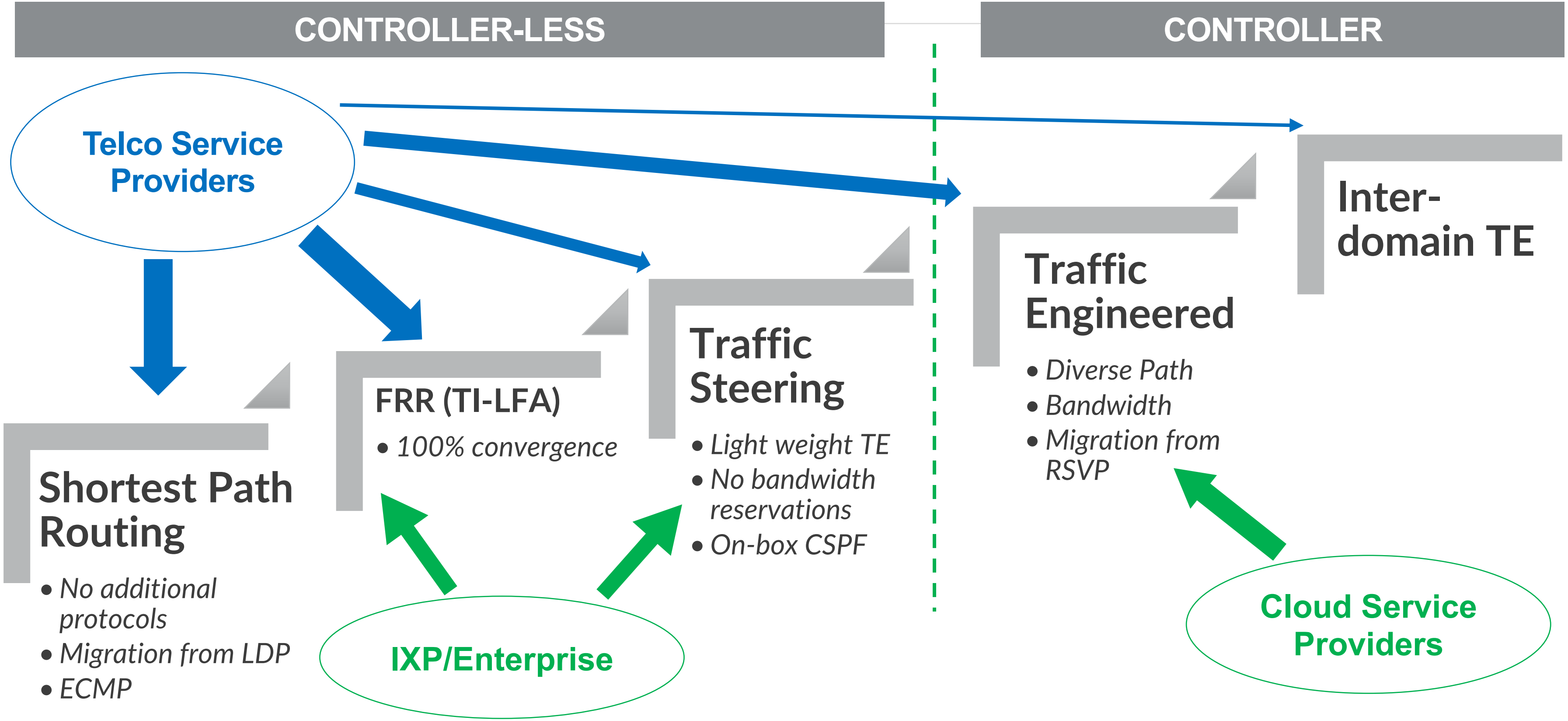
← Node-SID label to egress PE

← Node-SID label to egress PE

SPRING BENEFITS AND USECASES



SEGMENT ROUTING USE CASES



SEGMENT ROUTING FEATURE MAP

	Aggregation (LDP replacement)	Edge/Core (LDP/RSVP replacement, FRR)	IXP (Controller-less TE)	Cloud/Cable WAN (Controller TE)
Base SIDs	✓	✓	✓	✓
Traffic Statistics	✓	✓	✓	✓
LDP Interworking	✓	✓		
Services (L3VPN, EVPN, L2VPN, PW,…)	✓	✓	✓	✓
TI-LFA		✓	✓	✓
Static SR-TE		✓	✓	✓
RSVP coexistence (bandwidth)		✓	✓	✓
BGP SR-TE		✓		✓
PCEP SR-TE		✓		✓
EPE		✓		✓
BGP-LS		✓		✓
OAM	✓	✓	✓	✓
On-box CSPF for SR-TE		✓	✓	
NorthStar		✓		✓
Inter-AS		✓		✓

SEGMENT ROUTING (SPRING) BENEFITS

SR Use Case	Technical Benefit	Business Benefit
Traffic Engineering (TE)	Simpler TE using stateless core, eliminating the need for complex RSVP-TE and complex TE configurations	Network and operational simplicity translates into lower CAPEX & OPEX. Automated controller implementation of on-demand SR TE policy can unlock new business opportunity to sell more SLA-based services.
Traffic protection/Fast Reroute based on TI-LFA	100% coverage without increasing network statefulness, without micro loops, protects against all common failures (link, node, SRLG). Better than any other protection today.	Increase in network robustness and resilience. Faster convergence time and increased network availability.

SEGMENT ROUTING (SPRING) BENEFITS (CONTD.)

SR Use Case	Technical Benefit	Business Benefit
Network Programmability	Rich network programmability via centralised controller, using different kinds of labels (Node SID, Adj SID, Anycast SID, Binding SID) and deep label stacks	Opens the network for innovation and new services beyond just connectivity. OPEX and CAPEX reduction by having uniform transport layer across access, metro, core and DC. Operational simplicity without the need for additional tunneling protocol. Network slicing through inherent ability to TE and network programmability. Automatic traffic decisions lead to lower OPEX. Applications-based traffic control on low latency, high bandwidth across access, core and DC.

SEGMENT ROUTING (SPRING) BENEFITS (CONTD.)

SR Use Case	Technical Benefit	Business Benefit
Software Defined Networking and Network Function Virtualization	Flexibility to use SR in distributed, centralized and hybrid environments	Automatic traffic decisions lead to lower OPEX. Applications-based traffic control on low latency, high bandwidth across access, core and DC. Simple implementation of NFV service chaining.
End-to-end SR	Uniform SR transport layer across access, metro, core and DC eliminates the need for complex traffic re-inspection/reclassification at network layer boundaries.	OPEX and CAPEX reduction by having uniform transport layer across access, metro, core and DC
SRv6 as 5G Transport	SRv6 has a potential to replace mobile-specific tunneling protocols such as GTP-U.	Operational simplicity without the need for yet additional tunneling protocol. Network slicing through inherent ability to TE and network programmability.

KEY BENEFITS OF SEGMENT ROUTING VERSUS MPLS WITH LDP/RSVP-TE (CONTD.)

Benefit	Segment Routing	LDP	RSVP-TE
Operational Simplicity	Very simple	Simple	Complex
Fast Reroute	Yes, 100% coverage, with minimal traffic tromboning during FRR in ring topologies	Yes, <100% with LDP [r]LFA Yes, 100% with LDP TI-FRR at the expense of yet another stateful protocol (RSVP)	Yes, 100% with RSVP bypasses, much larger tromboning inevitable during FRR in ring topologies Yes, 100% with RSVP detours, minimal tromboning in ring topologies at the expense of much more state
Number of Protocols to test/configure/tune/maintain	IGP	IGP+LDP IGP+LDP+RSVP (in case of TI-FRR)	IGP+RSVP
SDN	Supported	Not supported	Only RSVP-TE supports SDN use cases

KEY BENEFITS OF SEGMENT ROUTING VERSUS MPLS WITH LDP/RSVP-TE

Benefit	Segment Routing	LDP	RSVP-TE
TE	Yes, simple	TE is not supported	Complex
ECMP for TE	Takes advantage of and automatically load-shares	Need to be specifically configured (one LDP session per ECMP link) to be able to take advantage of ECMP	Need to be specifically configured (at least one RSVP LSP per ECMP link) to be able to take advantage of ECMP
TE scalability	High: A minimal state (IGP state) in the core network is required. IGP state + TE state in the edge/source node is required	TE is not supported	Low: In addition to IGP state, RSVP needs to create & maintain ingress RSVP-TE LSP in edge nodes and core noes need to keep state of transit RSVP-TE LSPs

SR SCALABILITY ADVANTAGE

SR uses extensions to existing IGP protocols for signaling purpose.

SR is scalable because it does not rely on LDP/RSVP-TE, and there is no need to keep thousands of dynamically assigned session:label(s) records in an separate database.

- The SR Node SIDs are typically not changing and can only become known, not known (when the connectivity breaks), or known via alternative path.

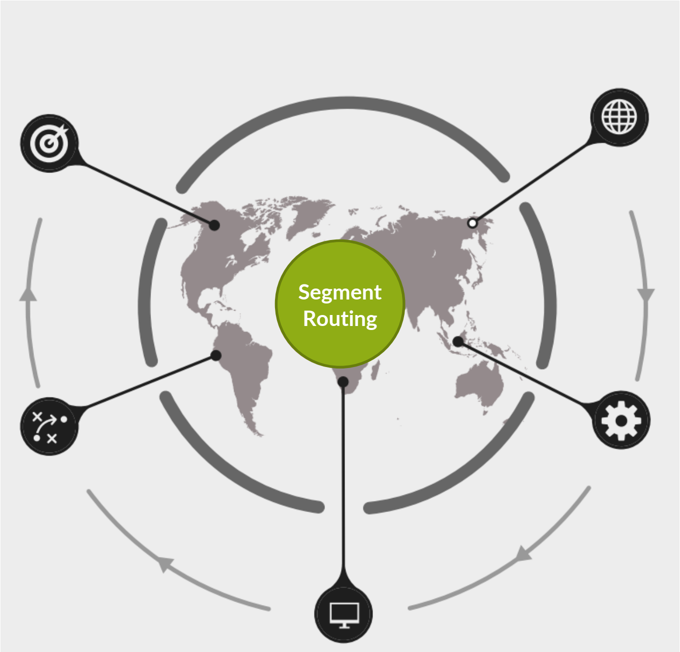
This SR property avoids thousands of MPLS TE LSPs in the network.

Relying on IGP has other benefits too, such as - it can automatically take advantage of Equal Cost Multi-Path (ECMP) routes to load balance across multiple available ECMP paths in the network and gain better bandwidth utilization. This kind of flexibility does not exist in LDP or RSVP-TE, which would need manual configurations to load-share traffic over ECMP paths.

SR BENEFITS: RECAP

Simplicity
Simple to operate, maintain and troubleshoot

Scalability
Scalable as the network core does not keep any state information allowing the core to scale further

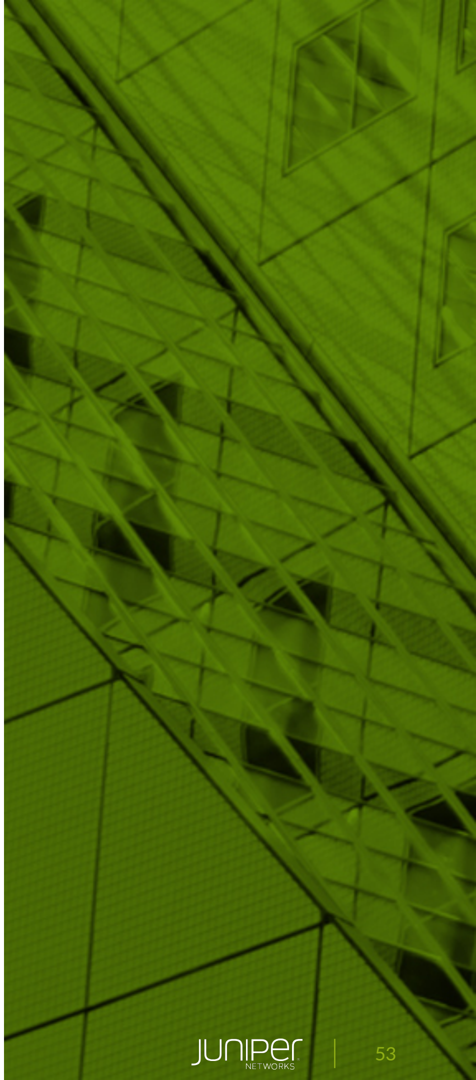


Fast Reroute
50msec can be guaranteed. Protection in all cases: link, node, srlg

Traffic Engineering
Complete control over how the traffic is routed in distributed or centralized control environment

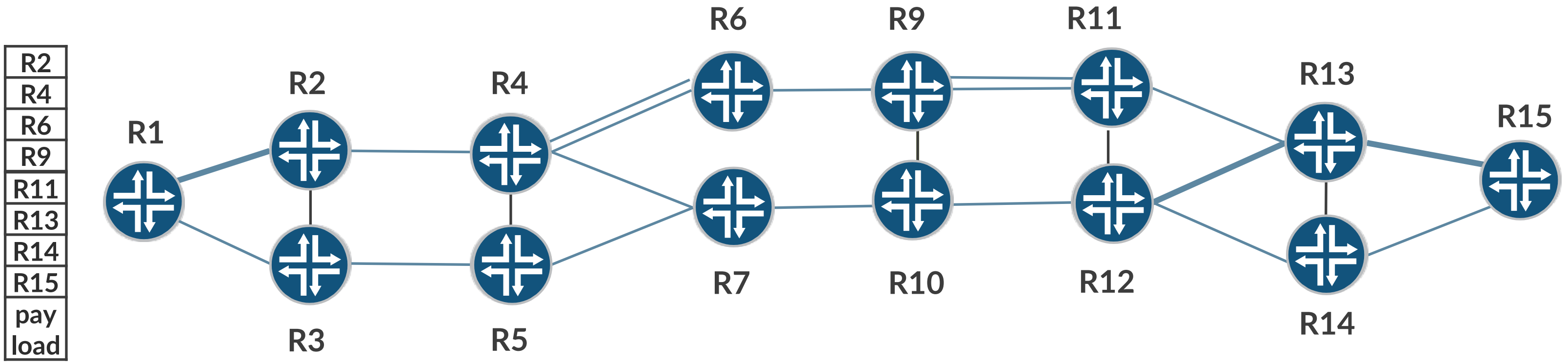
Network programmability
SR takes routing to the next level by bringing network programmability

DEPLOYMENT STRATEGY AND CONSIDERATIONS



LABEL STACK DEPTH CONSIDERATIONS FOR SR-TE

SR-TE requires consideration of label push and read depth



SR-TE SID depth

[edit protocols source-packet-routing]

```
maximum-segment-list-depth 16;
```

[edit protocols pcep pce <name>]

```
max-sid-depth 16;
```

SHIPS IN THE NIGHT

SR can co-exist with RSVP and LDP and BGP-LU

- Priorities RSVP → LDP → SR → BGP-LU
- If both classic LSP and SR LSR desired – different FEC (2nd IP on loopback).
  - Same as concurrent active LDP and RSVP in classic MPLS.
  - Service specific dataplane- MPLS or SR
- Easy migration. If old node can't be upgraded to run SPRING
  - Run SPRING on new nodes
  - Integrate Old Nodes using Spring Migration Tools- LDP Mapping Server/LDP Interworking/LDPoverSR-TE
  - RSVP-TE uses Spring Forwarding

Node-SID LSP could be non-continuous !
JUNOS implementation.
Early IOS TDP and LDP was like this.
https://youtu.be/T653QcGH8j8

OPTION 1: SHIPS IN THE NIGHT

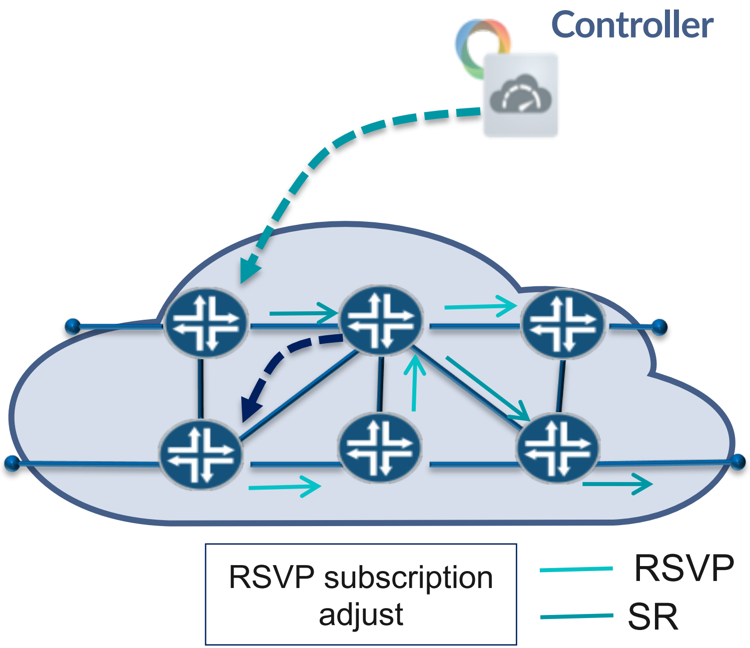
Concept

- Partition the network
  - RSVP max-reservable-bandwidth (subscription %)
  - Remaining is for SPRING traffic
- No changes required
  - Short-term goal to migrate to SPRING
- May lead to inefficient network usage

OPTION 2: SPRING AND RSVP NETWORK

Concept:

- SPRING traffic governs RSVP bandwidth allocations
- Each router measures SPRING traffic
- Backpressure RSVP based on configured thresholds
- Seamless: no modification to existing RSVP behavior



SPRING AND MIGRATION FROM LDP

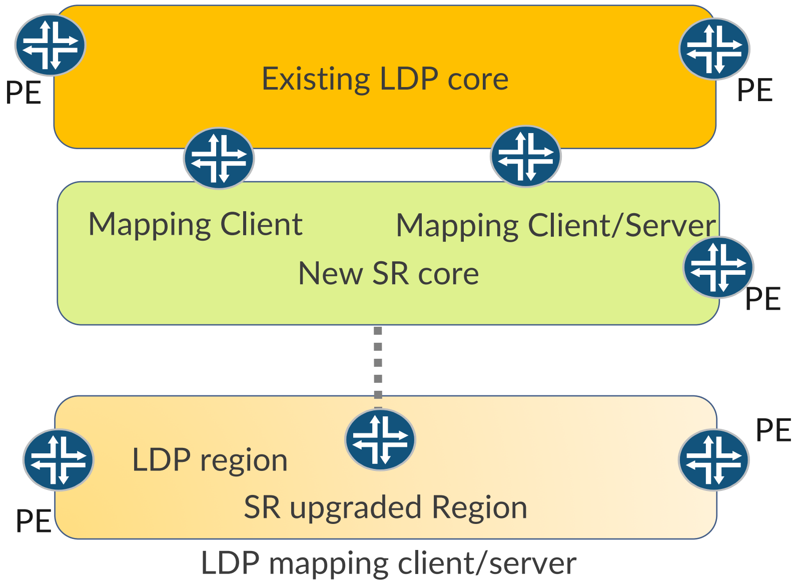
SPRING in a LDP Network

Concept

- Mapping Server creates FEC mappings and distributes in the SPRING network
- Mapping client stitches traffic between the domains

Considerations

- SRGB management
- Where to put Mapping Server
- Handling Mapping Label Conflicts
- Services mapped to either LDP or SPRING based transport

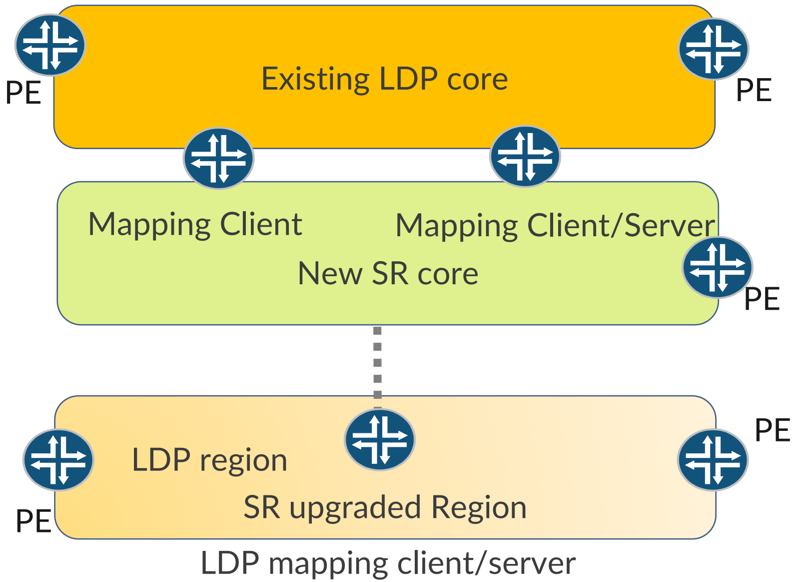


REDUNDANCY AND CONFLICTS IN SR-LDP MIGRATION

Redundancy for SID/label bindings via multiple mapping servers

Conflict resolution not supported for inconsistent advertisement from different SRMS

- ISIS
    1. Smaller range size is preferred.
    2. Lower index is preferred.
- OSPF
    - Lower router-id is preferred.



OAM: LSP PING AND TRACEROUTE

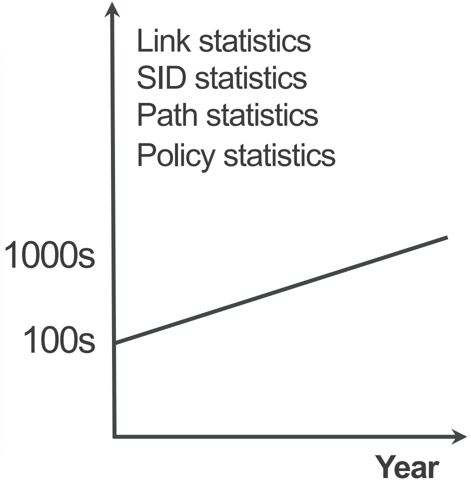
Segment Routing requires a robust OAM toolkit to report liveliness of network paths.

Support RFC 8287 for IPv4

- L-ISIS and L-OSPF
- ECMP path traceroute
- SR over RSVP supported
    - IETF draft: https://datatracker.ietf.org/doc/draft-arora-mpls-spring-ttl-procedures-srte-paths/

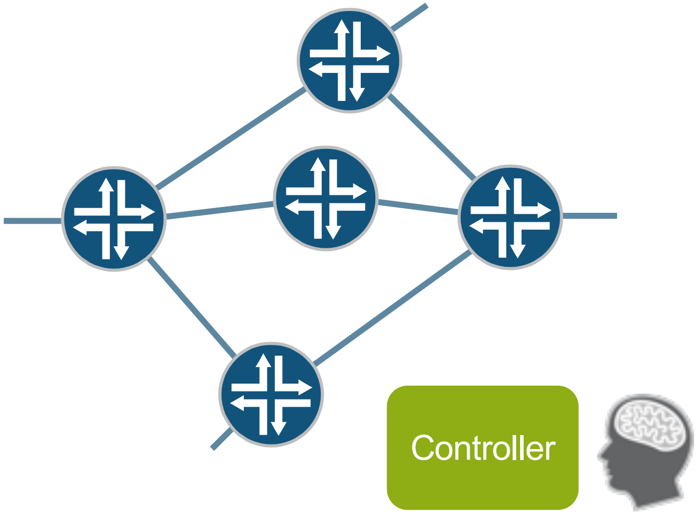
TRAFFIC STATISTICS TELEMETRY

Performance Monitoring, Accounting



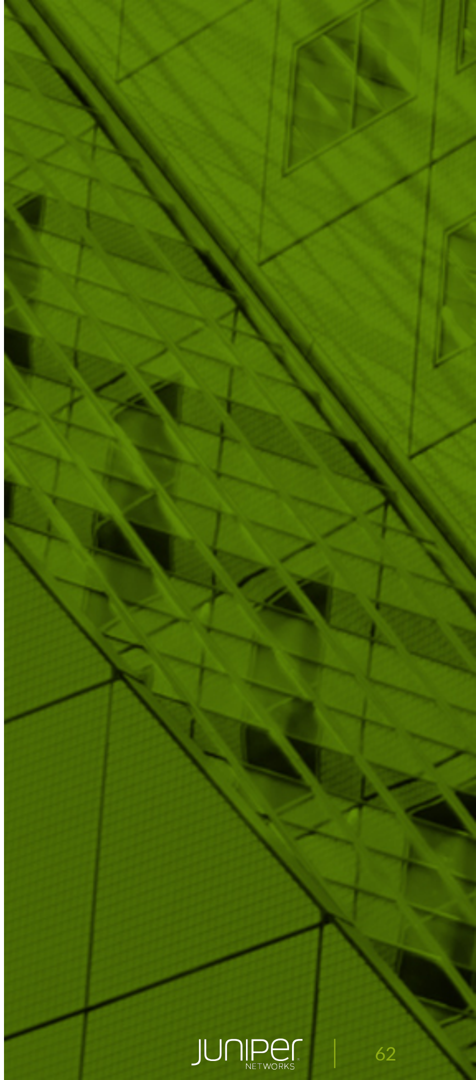
Traffic statistics for SR, SR-TE

Enabling External Control



State synchronization and tight feedback loop is required

SPRING CONFIGURATION AND VERIFICATION – SR/LDP INTERWORKING



SR/LDP INTERWORKING: CONFIGURATION

- SR Mapping Server (SRMS):

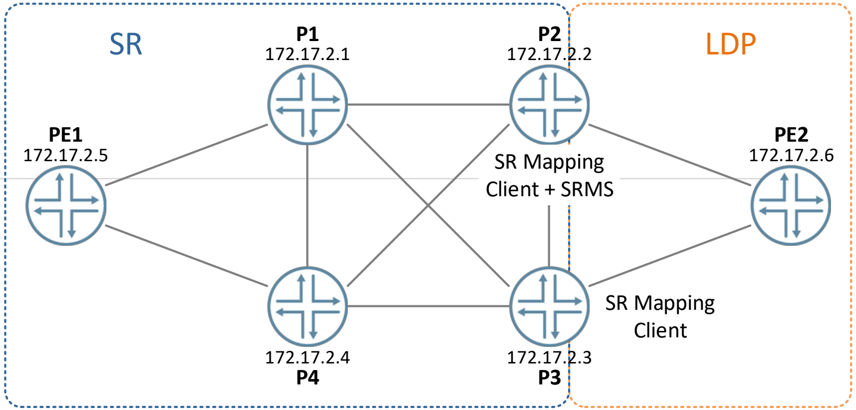
```
routing-options {
    source-packet-routing {
        mapping-server-entry p2map1 {                    ← defines SRMS map
            prefix-segment 172.17.2.6/32 index 6;        ← maps index to prefix for non-SR capable nodes
        }
    }
}
protocols {
    isis {
        source-packet-routing {
            mapping-server p2map1;                       ← applies SRMS map
        }
    }
}
```

SR/LDP INTERWORKING: CONFIGURATION

- SR Mapping Client:

```
protocols {
    isis {
        source-packet-routing {
            ldp-stitching;                    ← enables SR mapping client functionality in IS-IS
        }
    }
}
protocols {
    ldp {
        sr-mapping-client;                    ← enables SR mapping client functionality in LDP
    }
}
```

SR/LDP INTERWORKING: SR MAPPING SERVER

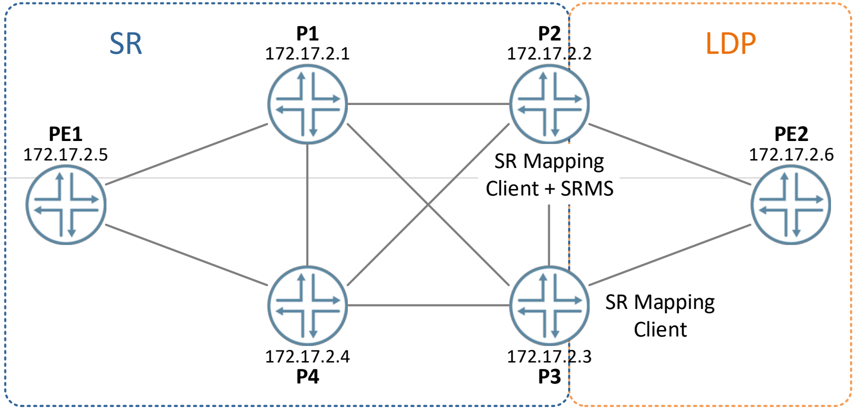


- SRMS (P2) advertises Node-SID on behalf of non-capable node (PE2)
  - Using IS-IS SID/Label Binding TLV

```
root@vmxdockerlight_p2_1> show isis database vmxdockerlight_p2_1.00-00 extensive
IS-IS level 2 link-state database:

vmxdockerlight_p2_1.00-00 Sequence: 0x318, Checksum: 0xaba9, Lifetime: 960 secs
  <...>
  TLVs:
    <...>
    Label binding: 172.17.2.6/32, Flags: 0x00(F:0,M:0,S:0,D:0,A:0), Range 1
      Node SID, Flags: 0x40(R:0,N:1,P:0,E:0,V:0,L:0), Algo: SPF(0), Value: 6
```

F-Flag: Address Family flag – 0 = IPv4, 1 = IPv6
M-Flag: Mirror Context flag – if set, the advertised SID corresponds to a mirrored context
S-Flag: If set, TLV should be flooded across the entire routing domain, if not not set, TLV must not be leaked between levels
D-Flag: Set when the SID/Label Binding TLV is leaked from level-2 to level-1
A-Flag: Attached flag – set if the prefixes and SIDs advertised are directly connected to the originator

SR/LDP INTERWORKING: SR MAPPING SERVER



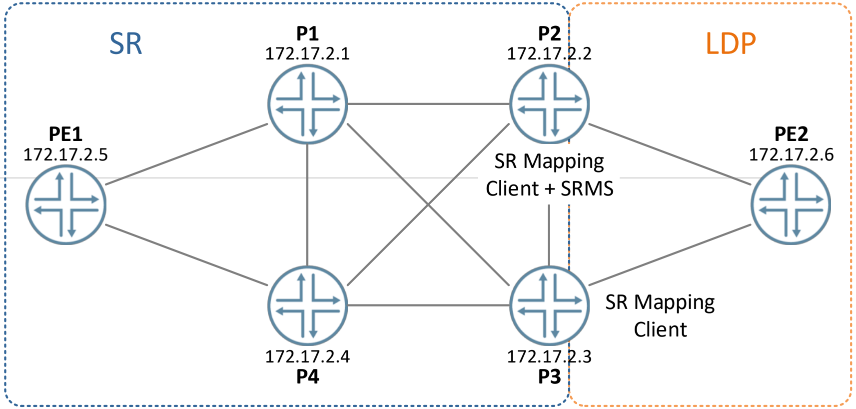
- SR-capable routers (PE1) install the advertised Node-SID (for PE2) in the data plane

```
root@vmxdockerlight_pe1_1> show route table inet.3 172.17.2.6

172.17.2.6/32      *[L-ISIS/14] 01:01:44, metric 40
                      to 172.22.66.3 via ge-0/0/0.0, Push 800006
                    > to 172.22.67.3 via ge-0/0/1.0, Push 800006
```

- Non SR-capable routers (PE2) use the LDP labels

```
root@vmxdockerlight_pe2_1> show route table inet.3 172.17.2.5

172.17.2.5/32      *[LDP/15] 00:07:12, metric 40
                      to 172.22.70.3 via ge-0/0/0.0, Push 91
                    > to 172.22.71.3 via ge-0/0/1.0, Push 63
```

SR/LDP INTERWORKING: SR MAPPING CLIENT



- SR mapping client (P2, P3) swaps the node segment label and the LDP label
- LDP-to-SR traffic flow (right to left):

```
root@vmxdockerlight_p2_1> show route table mpls.0 label 91
91                    *[LDP/15] 00:08:54, metric 1
                       > to 172.22.64.3 via ge-0/0/0.0, Swap 801005
                         to 172.22.73.3 via ge-0/0/2.0, Swap 801005

root@vmxdockerlight_p3_1> show route table mpls.0 label 63
63                    *[LDP/15] 00:08:15, metric 1
                       > to 172.22.72.2 via ge-0/0/0.0, Swap 801005
                         to 172.22.68.2 via ge-0/0/2.0, Swap 801005
```

- SR-to-LDP traffic flow (left to right):

```
root@vmxdockerlight_p2_1> show route table mpls.0 label 800006
800006                *[L-ISIS/14] 00:21:52, metric 20
                       > to 172.22.70.2 via ge-0/0/3.0, Pop
```

TELEMETRY (JTI): CONFIGURATION

```
services {
    analytics {
        streaming-server host {
            remote-address 172.22.64.1;
            remote-port 50000;
        }
        export-profile host-local {
            local-address 172.22.64.3;
            local-port 1000;
            reporting-rate 2;
            payload-size 2000;
            format gpb;
            transport udp;
        }
        sensor srte-egress {
            server-name host;
            export-name host-local;
            resource /junos/services/segment-
                    routing/interface/egress/usage/;
        }
        sensor srte-ingress {
            server-name host;
            export-name host-local;
```

```
            resource /junos/services/segment-
                    routing/interface/ingress/usage/;
        }
        sensor srte-sid {
            server-name host;
            export-name host-local;
            resource /junos/services/segment-
                    routing/sid/usage/;
        }

protocols {
    source-packet-routing {
        statistics {
            telemetry;
        }
    }
    isis {
        source-packet-routing {
            sensor-based-stats {
                per-interface-per-member-link
                                   ingress egress;
                per-sid ingress;
            }
```

TELEMETRY (JTI): SAMPLE EXPORTS

```
system_id: "vmxdockerlight_p1_1:172.22.66.2"
component_id: 0
sensor_name: "srte-ingress:/junos/services/segment-
routing/interface/ingress/usage/:/junos/services/segme
nt-routing/interface/ingress/usage/:PFE"
sequence_number: 36
timestamp: 1527268196431
version_major: 1
version_minor: 0
enterprise {
  [juniperNetworks] {
    [jnpr_sr_stats_per_sid_ext] {
      sid_stats {
        sid_identifier: "801005"
        instance_identifier: 0
        counter_name: "oc-22"
        ingress_stats {
          packets: 3823
          bytes: 420530
          packet_rate: 43
          byte_rate: 4826
        }
      }
```

```
    [jnpr_sr_stats_per_if_ingress_ext] {
      per_if_records {
        if_name: "ge-0/0/0.0"
        ingress_stats {
          packets: 711
          bytes: 78210
          packet_rate: 43
          byte_rate: 4817
        }
      }

    [jnpr_sr_stats_per_if_egress_ext] {
     per_if_records {
        if_name: "ge-0/0/3.0"
        counter_name: "oc-6"
        egress_stats {
          packets: 3919
          bytes: 431090
          packet_rate: 43
          byte_rate: 4787
        }
      }
```

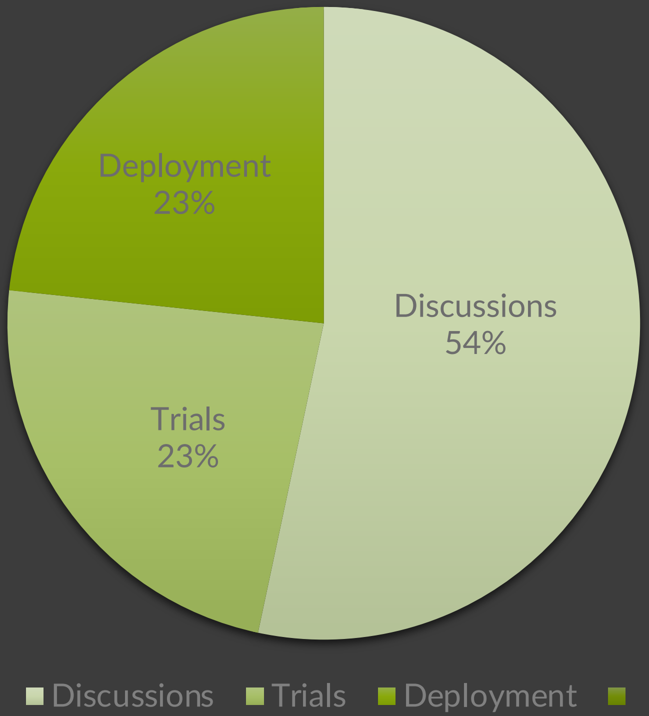
SPRING CUSTOMER ENGAGEMENTS

30+ Customer Engagements

16+ Customer Discussions

7+ Customer POC/Trial

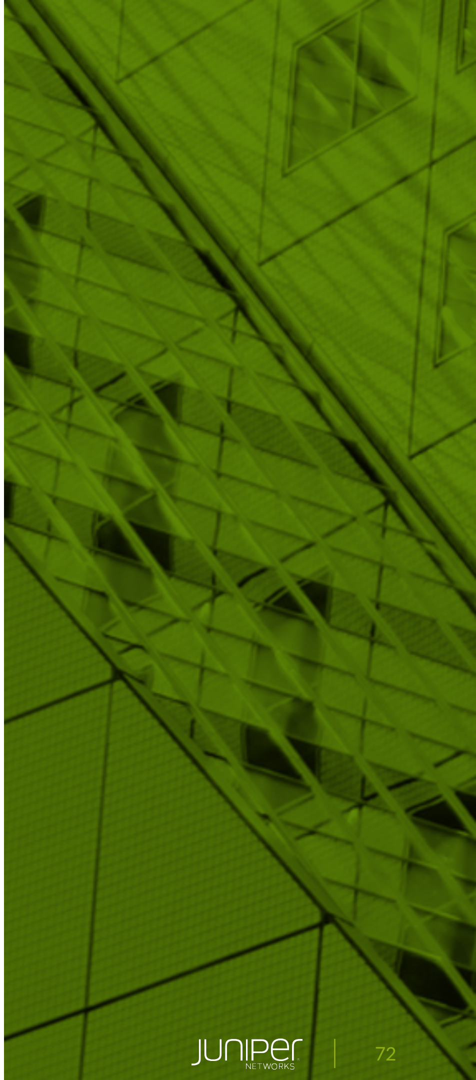
7+ Customer Deployments



SPRING IN SERVICE PROVIDERS' NETWORKS: PUBLIC REFERENCES

Company name	Use case	Drivers for Selecting Segment Routing	Benefits
China Unicom	Deployment in the backbone with SDN controller	Making network ready for cloud. China Unicom migration to cloud can only be achieved by having consistent and simple protocol across multiple domains	Elimination of complex protocols from backbone Centralized PCE based controller enables China Unicom to offer cloud based services
Colt	Deployment in Colt IQ network across Pan European, US and Asian packet network	Combined SR and EVPN, to offer faster convergence, increased network availability and resiliency for any topology	Simplify and automate network operations and reduce OPEX
Vodafone Germany	Traffic engineering in MPLS Core	Ability to engineer paths based on latency and application requirements	Simplified Operations 50% latency reduction

SPRING TRAFFIC ENGINEERING



WHAT IS TRAFFIC ENGINEERING?

- Any time we want traffic to follow a path that is not the shortest path, as computed by our IGPs
  - Resource optimization
  - Disjoint path
  - Regulatory
  - Application performance, …

- What is the customer use-case for TE?

- A label stack in SR-TE implies TE
- Computing the label stack to include TE constraints (e.g. admin-colors).
- Computation can be on-box (CSPF) or off-box (controller).

SR-TE INTRODUCTION

- Simple, Automated and Scalable
  - No Core State : state is in the packet header
  - No tunnel interface : “SR Policy”
  - On Demand policy instantiation
  - Automated Steering of packets

- Multi-Domain
  - SR Controller
  - Binding-SID for stitching multiple segments

- SRTE architecture applies to MPLS and IPv6 applications

- SRTE next-hop is a list or lists of SIDs that operator wants incoming traffic to use.

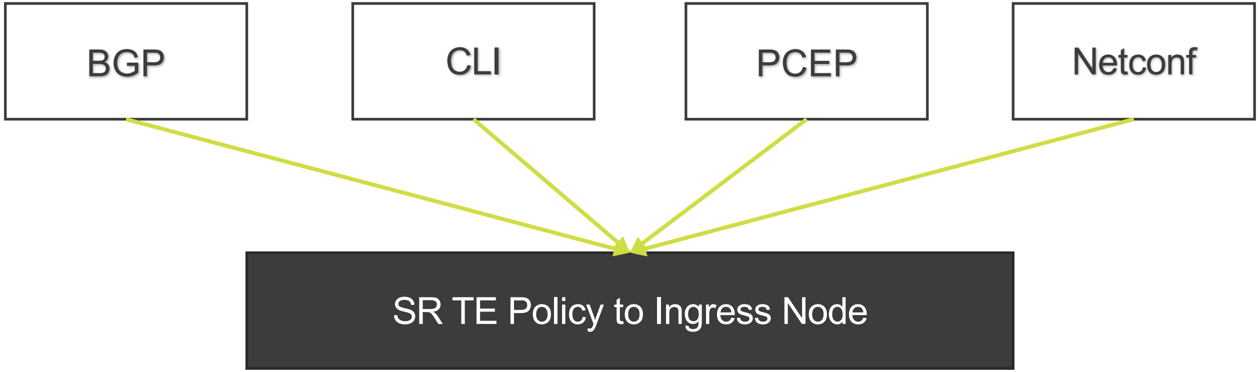
How are SR-TE LSPs instantiated on a Junos router?

- Static Configuration
- BGP SR-TE (from a controller)
- PCEP (from a controller)
- CSPF computation on ingress node

SR-TE PATH PROGRAMMING

Path programming options

- A head-end node can get to know about SR-TE path for a SR Policy by various means.
- SR-Policy is represented in FIB as a BSID-keyed entry
- A path is selected for a SR Policy when it is valid & its preference is the highest value
- The protocol source of the path does not matter in path selection logic
- When a new Cpath is learned or previous Cpath expires, selection process is re-executed
- SRTE policy maintains a SR-TE Database that is multi-domain capable



SR-TE WITH CONTROLLER
TOPOLOGY DISCOVERY AND PROVISIONING

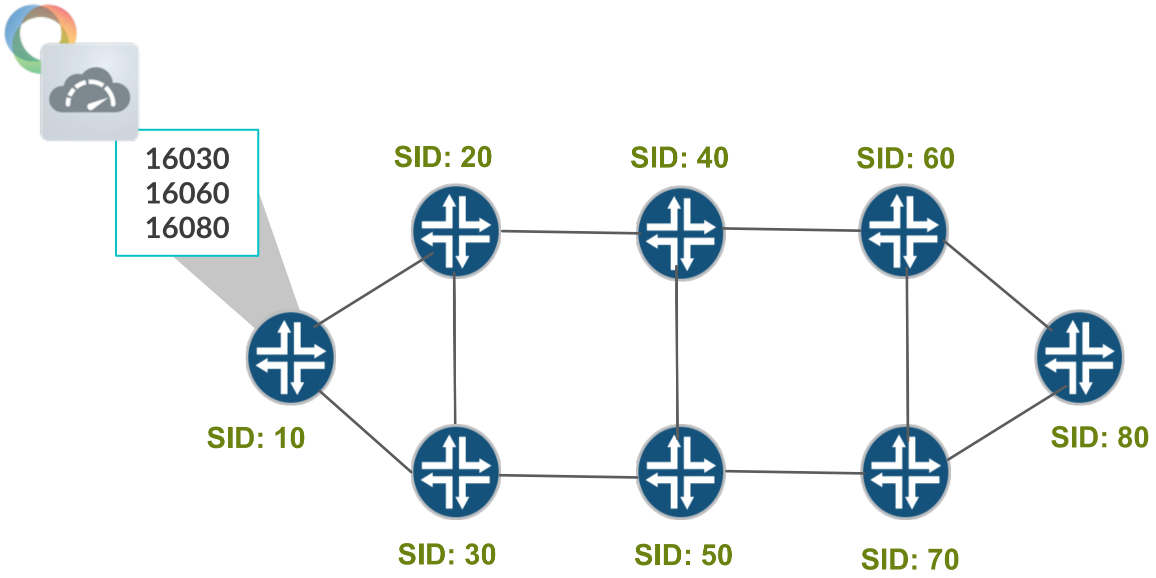
Components:

Topology discovery

- IGP
- BGP-LS
- IGP DB export (gRPC)

Path provisioning – TE Policy

- BGP-LU
- BGP SR-TE
- PCEP
- Netconf/Yang
- Programmable interface (gRIBI)



SR-TE CONSTRAINTS AND LABEL STACK

Path computation:

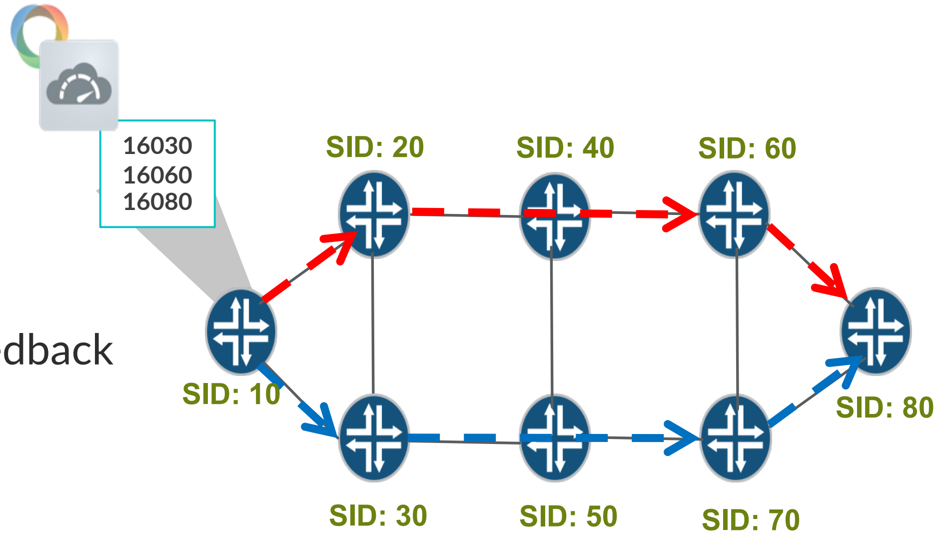
- Constraints: admin-colors, …
- On-box vs. TE controller
- Bandwidth: Requires a controller

Telemetry:

- Optimization requires a tight statistics feedback loop

Label stack having node-SID:

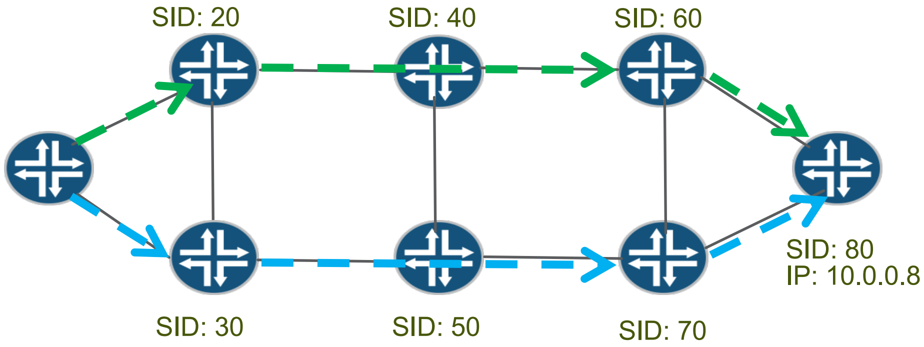
- Allows for ECMP in core via IGP paths
- Possible only if constraints allow for picking the node-SID
- Beware of traffic taking unintended links



STATIC SR-TE

Concept

- Named LSP
- Configured as SR-TE Policy
- Can have 1 or more segment lists
- Persistent LSP state
- Binding-SID
- W-ECMP
- Secondary path support
- Label translation service for IPv4 addresses




```
protocols spring-traffic-engineering {
  segment-list P1 {
    hop1 20
    hop2 40
    hop3 60
…
  }
}
 segment-list P2 {
 etc
```

```
source-routing-path sr_lsp1 {
                    to 10.0.0.8
                    color 10
                    binding-sid 1000 (optional)
                    primary P1 weight 2
                    primary P2 weight 1
                    }}
```

SEAMLESS BFD – RAPID FAILURE DETECTION IN SR-TE

- Supported for **static SR-TE LSP** for first hop label
  - Colored and Uncolored
  - S-BFD runs for each segment list of SR-TE LSP (shared if same label stack)
  - Segment list is included in forwarding once S-BFD comes up
- Failure actions for S-BFD down at ingress
  - RE receives trigger
    - Removes segment-list forwarding from SR-TE
    - If last segment-list down, then switches to secondary SR-TE policy
    - Reversion from secondary to primary after SBFD comes up on primary
  - S-BFD detection and repair not inline distributed to line cards

ISIS: 17.4

BGP SPRING TE

Concept:

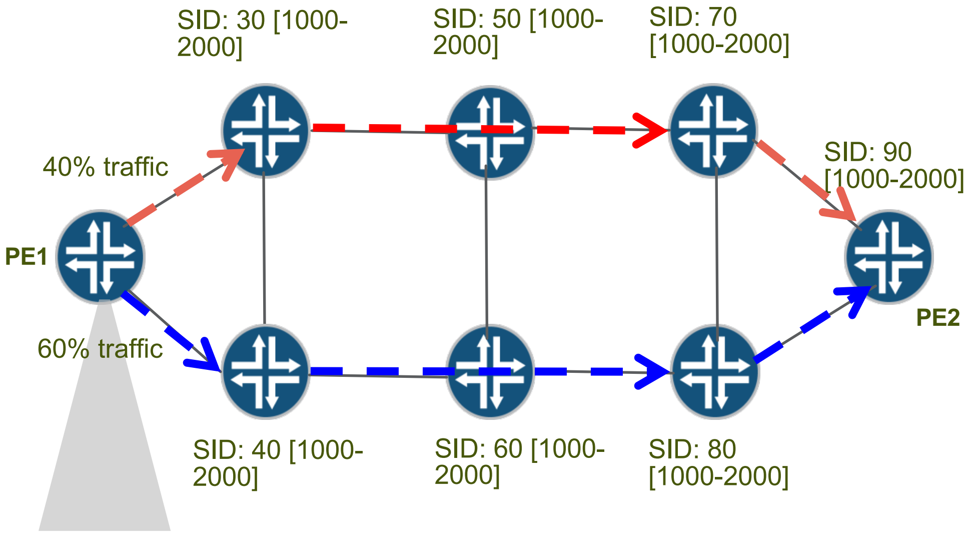
- Similar concept as static, but conveyed via BGP instead of via CLI config
- New BGP NLRI

Use-cases:

- Centralized Traffic Engineering

Example:

- Traffic load-balanced in the ratio 40:60 (Weighted ECMP)



BGP TE – policy
Binding SID: 1000
[40% Red – 1030, 1050, 1070, 1090],
[60% Blue – 1040, 1060, 1080, 1090]

TE WITHOUT CONTROLLER - DYNAMIC SR-TE

Requirements

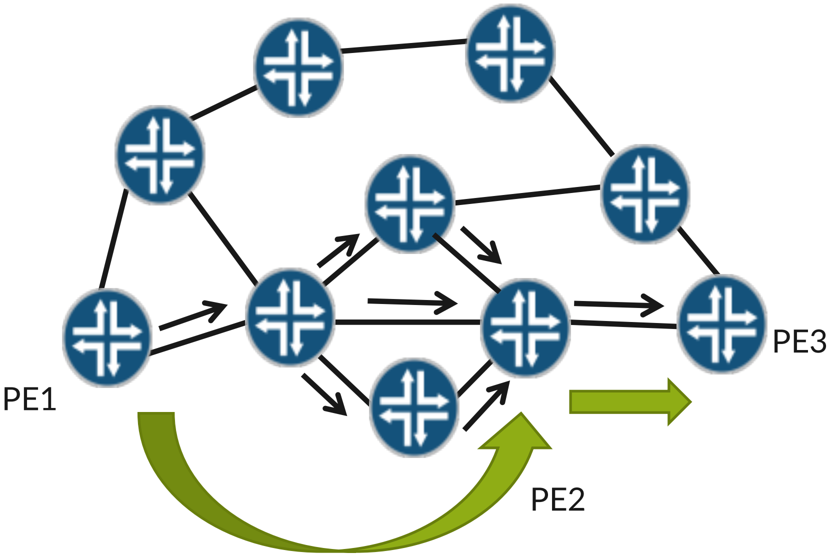
- Enable traffic engineering use-case without centralized controller
  - No bandwidth reservation requirement
  - Lightweight TE based on link constraints
  - Decouple TE attribute flooding from RSVP

Concept

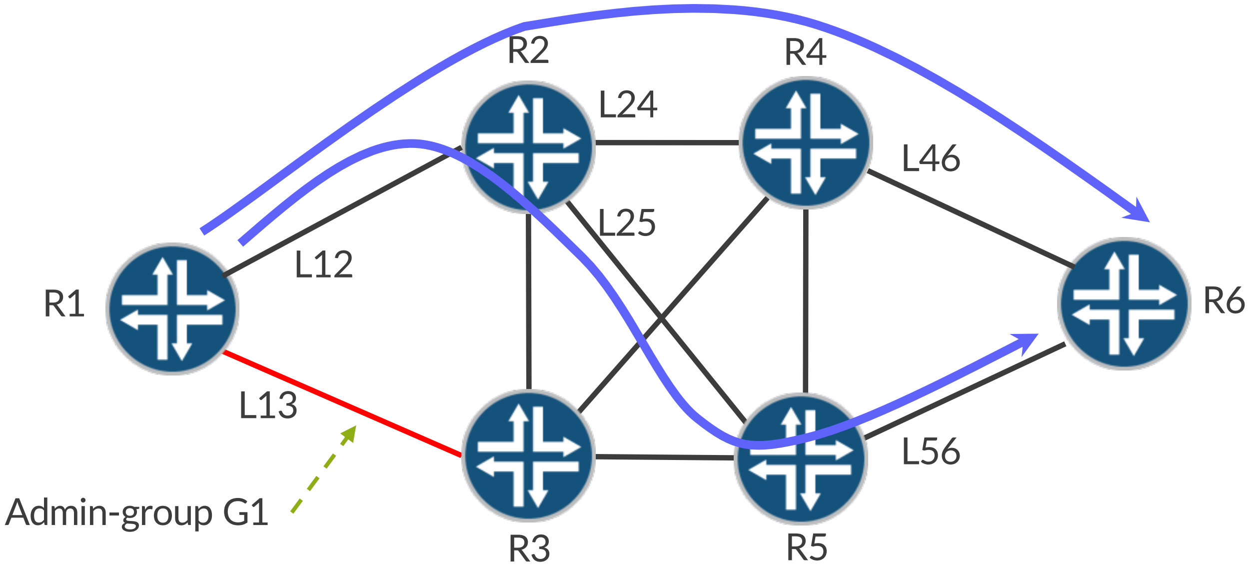
- Path computation is done on ingress router
  - TE constraints (e.g. admin colors)

Computation results

- Strict hops (stack of Adj-SIDs)
- SR native algorithm (compression with node-SID)



SPRING CSPF EXAMPLE 1

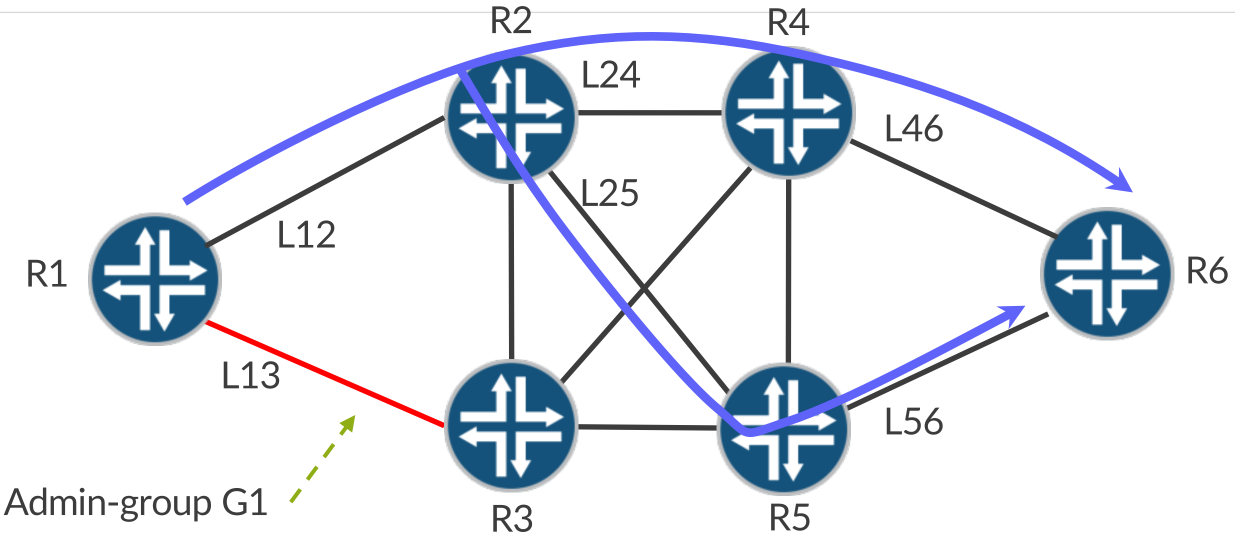


Configured on R1:

To R6
Exclude G1
No-label-stack-compression

Outcome: two paths are computed and installed, each consisting of adjacency SIDs:
{L12, L24, L46}
{L12, L25, L56}

SPRING CSPF EXAMPLE 2

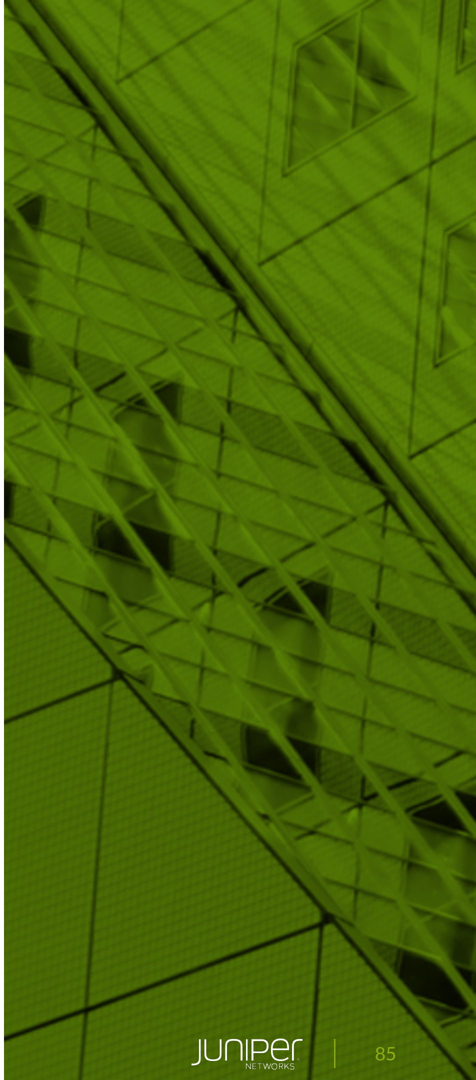


Configured on R1:

To R6
Exclude G1

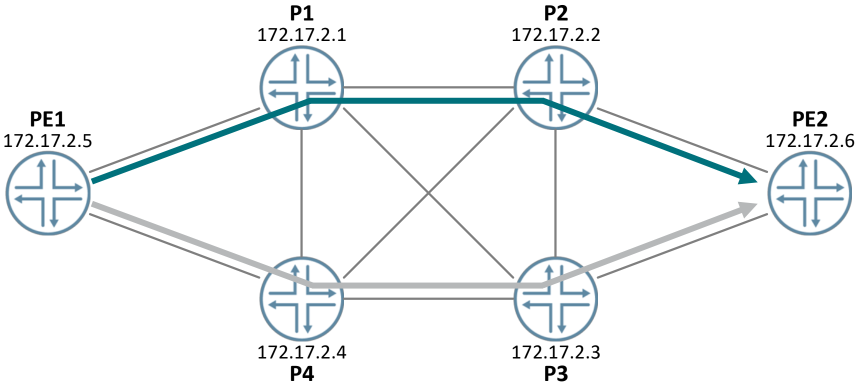
Outcome: path consisting of stack of node-SIDs is computed and installed: {R2, R6}.
Path has inherent ECMP between R2 and R6.

SPRING CONFIGURATION AND VERIFICATION – TRAFFIC ENGINEERING



STATIC SR LSPS (UNCOLORED) – INGRESS PE EXAMPLE: CONFIGURATION

```
protocols {
    source-packet-routing {
        segment-list P1-P2-PE2 {
            P1 ip-address 172.22.66.3;
            P2 label 801002;
            PE2 label 800006;
        }
        segment-list P4-P3-PE2 {
            P4 ip-address 172.22.67.3;
            P3 label 801003;
            PE2 label 800006;
        }
        source-routing-path NORTH {
            to 172.17.6.10;
            primary {
                P1-P2-PE2;
            }
        }
        source-routing-path SOUTH {
            to 172.17.6.20;
            primary {
                P4-P3-PE2;
            }
```

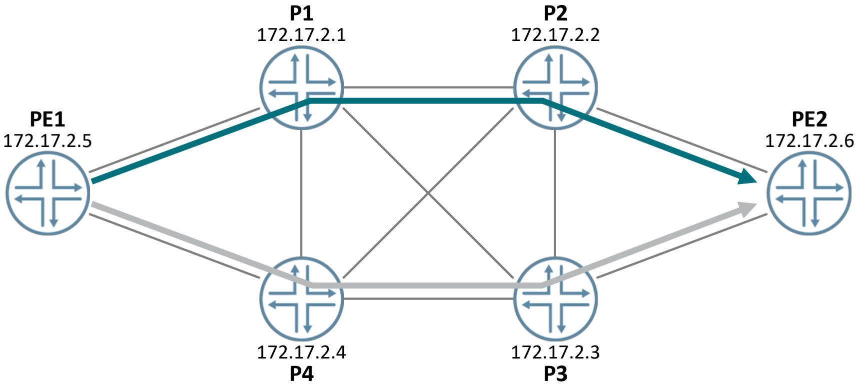


- A segment list can have one or multiple hops
- First hop must be an IP address to resolve OIF
- Destination address defines SR tunnel endpoint
- Support primary and secondary segment lists
- Supports multiple SR paths for the same destination with preferences

STATIC SR LSPS (UNCOLORED) – INGRESS PE EXAMPLE: VERIFICATION

- An ingress route for the LSP destination is installed in the inet.3 table

```
root@vmxdockerlight_pe1_1> show route table inet.3

172.17.6.10/32      *[SPRING-TE/8] 00:13:12, metric 1
                     > to 172.22.66.3 via ge-0/0/0.0, Push 800006, Push 801002(top)
172.17.6.20/32      *[SPRING-TE/8] 00:13:12, metric 1
                     > to 172.22.67.3 via ge-0/0/1.0, Push 800006, Push 801003(top)
```



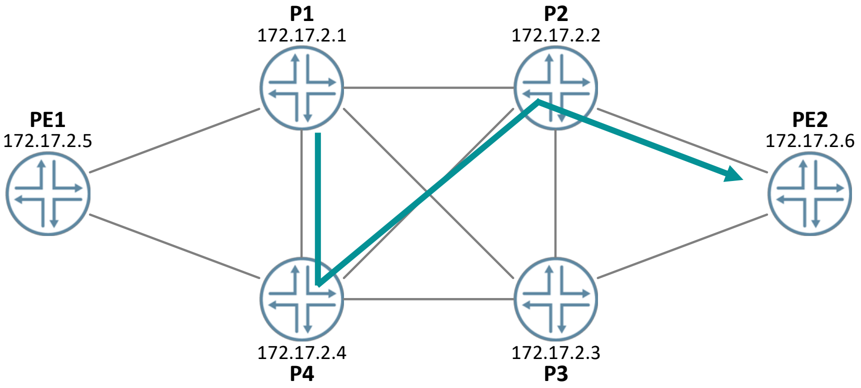
STATIC SR LSPS (UNCOLORED) – INGRESS PE EXAMPLE: VERIFICATION

- Routes in table inet.3 can be used to resolve BGP routes

```
root@vmxdockerlight_pe1_1> show route protocol bgp detail
inet.0: 27 destinations, 27 routes (27 active, 0 holddown, 0 hidden)

11.0.0.12/32 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                <...>
                Next hop: 172.22.66.3 via ge-0/0/0.0 weight 0x1, selected
                Label operation: Push 800006, Push 801002(top)
                <...>
                Protocol next hop: 172.17.6.10
                <...>
11.0.0.13/32 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                <...>
                Next hop: 172.22.67.3 via ge-0/0/1.0 weight 0x1, selected
                Label operation: Push 800006, Push 801003(top)
                <...>
                Protocol next hop: 172.17.6.20
                <...>
```

STATIC SR LSPS (UNCOLORED) – TRANSIT P EXAMPLE: CONFIGURATION

```
protocols {
    source-packet-routing {
        segment-list P4-P2-PE2 {
            P4 ip-address 172.22.65.3;
            P2 label 801002;
            PE2 label 800006;
        }
        source-routing-path DETOUR-VIA-P4 {
            to 172.17.6.10;
            binding-sid 1000610;
            primary {
                P4-P2-PE2;
            }
        }
    }
}
```

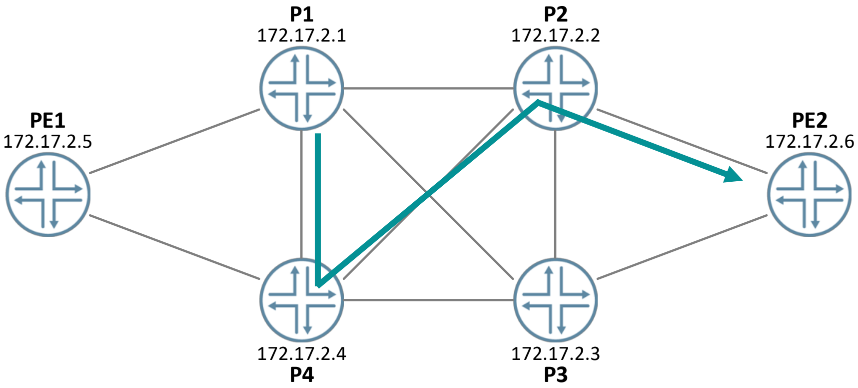


- Binding-SID label maps incoming labeled traffic to the SR path
- Binding-SID label can also be used to stitch multiple LSPs

STATIC SR LSPS (UNCOLORED) – TRANSIT P EXAMPLE: VERIFICATION

- For the Binding SID label a route is installed in the mpls.0 table

```
root@vmxdockerlight_p1_1> show route table mpls.0

1000610             *[SPRING-TE/8] 00:16:11, metric 1
                     > to 172.22.65.3 via ge-0/0/2.0, Swap 800006, Push 801002(top)
```



ROUTING POLICY EXAMPLE: CONFIGURATION

- Policy steers traffic into static uncolored SR LSPs e.g. based on communities

```
policy-options {
    policy-statement SR-POLICY1 {
        term 1 {
            from community SR-COMM3;
            then {
                install-nexthop lsp VIA-P1-P2;
                accept;
            }
        }
        term 2 {
            from community SR-COMM4;
            then {
                install-nexthop lsp VIA-P4-P3;
                accept;
            }
        }
    }
    community SR-COMM3 members target:65320:33;
    community SR-COMM4 members target:65320:44;
}
```

```
routing-options {
    forwarding-table {
        export SR-POLICY1;
    }
}
protocols {
    source-packet-routing {
        source-routing-path VIA-P1-P2 {
            to 172.17.2.6;
            primary {
                P1-P2-PE2;
            }
        }
        source-routing-path VIA-P4-P3 {
            to 172.17.2.6;
            primary {
                P4-P3-PE2;
            }
        }
    }
```

ROUTING POLICY EXAMPLE: VERIFICATION

```
root@vmxdockerlight_pe1_1> show route detail 13.0.0.0/16 | match "entry|BGP|protocol|communities"

13.0.1.0/24 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Protocol next hop: 172.17.2.6
                Communities: target:65320:33
13.0.2.0/24 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                Protocol next hop: 172.17.2.6
                Communities: target:65320:44

root@vmxdockerlight_pe1_1> show route forwarding-table
Routing table: default.inet
Internet:
Destination        Type RtRef Next hop           Type Index    NhRef Netif
13.0.1.0/24        user     0                     indr  1048589     2
                              172.22.66.3        Push 800006, Push 801002(top)      610      2 ge-0/0/0.0
13.0.2.0/24        user     0                     indr  1048588     2
                              172.22.67.3        Push 800006, Push 801003(top)      614      2 ge-0/0/1.0
```

CLASS-BASED FORWARDING EXAMPLE: CONFIGURATION

- Policy steers traffic into static SR LSPs based on forwarding classes

```
routing-options {
    forwarding-table {
        export SR-CBF-POLICY;
    }
}
policy-options {
    policy-statement SR-CBF-POLICY {
        term 1 {
            from {
                route-filter 11.0.0.14/32 exact;
            }
            then {
                cos-next-hop-map SR-CBF-MAP1;
                accept;
            }
        }
    }
}
```

```
class-of-service {
    forwarding-policy {
        next-hop-map SR-CBF-MAP1 {
            forwarding-class BestEffort {
                lsp-next-hop VIA-P4-P2-P3;
            }
            forwarding-class LowLoss {
                lsp-next-hop VIA-P4-P2-P3;
            }
            forwarding-class LowDelay {
                lsp-next-hop VIA-P1-P3;
            }
            forwarding-class Voice {
                lsp-next-hop VIA-P1-P3;
            }
        }
    }
}
```

CLASS-BASED FORWARDING EXAMPLE: CONFIGURATION

```
protocols {
    source-packet-routing {
        segment-list P1-P3-PE2 {
            P1 ip-address 172.22.66.3;
            P3 label 801003;
            PE2 label 800006;
        }
        segment-list P4-P2-P3-PE2 {
            P4 ip-address 172.22.67.3;
            P2 label 801002;
            P3 label 801003;
            PE2 label 800006;
        }
        source-routing-path VIA-P1-P3 {
            to 172.18.2.4;
            primary {
                P1-P3-PE2;
            }
        }
        source-routing-path VIA-P4-P2-P3 {
            to 172.18.2.4;
            primary {
                P4-P2-P3-PE2;
            }
```

CLASS-BASED FORWARDING EXAMPLE: VERIFICATION

- BGP NH resolved via static SR LSP entry in table inet.3

```
root@vmxdockerlight_pe1_1> show route 11.0.0.14/32 detail | match "Protocol next hop"
                Protocol next hop: 172.18.2.4

root@vmxdockerlight_pe1_1> show route 172.18.2.4 table inet.3
inet.3: 11 destinations, 12 routes (11 active, 0 holddown, 0 hidden)

172.18.2.4/32      *[SPRING-TE/8] 00:30:05, metric 1
                      to 172.22.66.3 via ge-0/0/0.0, Push 800006, Push 801003(top)
                    > to 172.22.67.3 via ge-0/0/1.0, Push 800006, Push 801003, Push 801002(top)

root@vmxdockerlight_pe1_1> show route 11.0.0.14/32
inet.0: 34 destinations, 34 routes (34 active, 0 holddown, 0 hidden)

11.0.0.14/32       *[BGP/170] 1d 02:52:04, localpref 100, from 172.17.2.6
                      AS path: I, validation-state: unverified
                    > to 172.22.66.3 via ge-0/0/0.0, Push 800006, Push 801003(top)
                      to 172.22.67.3 via ge-0/0/1.0, Push 800006, Push 801003, Push 801002(top)
```

CLASS-BASED FORWARDING EXAMPLE: VERIFICATION

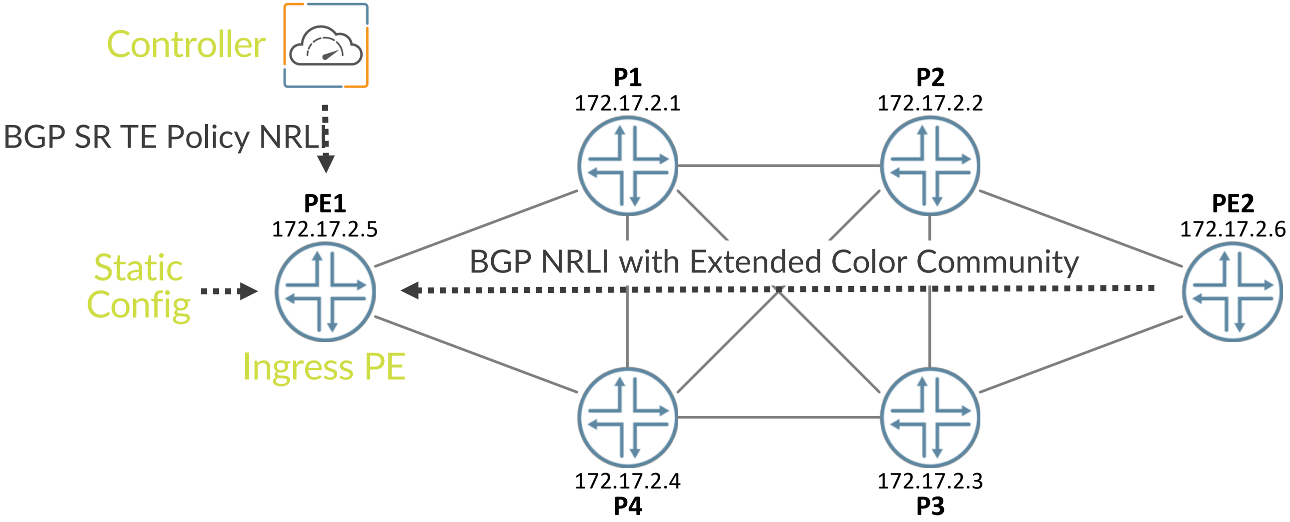
- Forwarding table entries with "idx" indicate CBF

```
root@vmxdockerlight_pe1_1> show route forwarding-table destination 11.0.0.14/32
Routing table: default.inet
Internet:
Enabled protocols: Bridging,
Destination      Type RtRef Next hop           Type Index     NhRef Netif
11.0.0.14/32     user     0                     indr  1048582     2
                                                idxd      599     2
                 idx:1      172.22.66.3        Push 800006, Push 801003(top)     604    3 ge-0/0/0.0
                 idx:2      172.22.66.3        Push 800006, Push 801003(top)     604    3 ge-0/0/0.0
                 idx:3      172.22.67.3        Push 800006, Push 801003, Push 801002(top)     593    3 ge-0/0/1.0
                 idx:xx     172.22.67.3        Push 800006, Push 801003, Push 801002(top)     593    3 ge-0/0/1.0
```

SR TE POLICY – SETUP

SR TE Policy injected via BGP or static configuration

Remote prefixes advertised via BGP with Extended Color Community (Type 0x03 / Sub-type 0x0b) to steer traffic into a policy



SR TE POLICY – BGP CONFIGURATION (INGRESS PE)

```
protocols {
    bgp {
        group iBGP {                                  ← BGP session(s) to remote PEs
            type internal;
            neighbor 172.17.2.6 {
                family inet {
                    unicast {
                        extended-nexthop-color;       ← enables Extended Color Community
                    }
                }
            }
        }
        group SR_TE {                                 ← BGP session(s) to controller
            type internal;
            family inet {
                unicast;
                segment-routing-te;                   ← enables SR TE Policy SAFI and NRLI
            }
            neighbor 172.22.63.254;
        }
    }
```

SR TE POLICY – ADVERTISING NRLI WITH EXTENDED COLOR COMMUNITY (REMOTE PE): CONFIGURATION

```
protocols {
    bgp {
        group iBGP {
            type internal;
            export LOCAL-ROUTES;
            neighbor 172.17.2.5 {
                family inet {
                    unicast {
                        extended-nexthop-color;
                    }
                }
            }
        }
    }
}
policy-options {
    policy-statement LOCAL-ROUTES {
        term COMM-COLOR-1111 {
            from {
                route-filter 11.0.1.11/32 exact;
            }
```

```
            then {
                community add SR-COMM1;
                accept;
            }
        }
        term COMM-COLOR-2222 {
            from {
                route-filter 11.0.1.22/32 exact;
            }
            then {
                community add SR-COMM2;
                next-hop 172.17.6.10;
                accept;
            }
        }
    }
    community SR-COMM1 members color:0:1111;
    community SR-COMM2 members color:0:2222;
}
```

SR TE POLICY – ADVERTISING NRLI WITH EXTENDED COLOR COMMUNITY (REMOTE PE): VERIFICATION

```
root@vmxdockerlight_pe2_1> show route advertising-protocol bgp 172.17.2.5 detail

inet.0: 29 destinations, 29 routes (29 active, 0 holddown, 0 hidden)

* 11.0.1.11/32 (1 entry, 1 announced)
 BGP group iBGP type Internal
     Nexthop: Self
     Localpref: 100
     AS path: [3320] I
     Communities: color:0:1111

* 11.0.1.22/32 (1 entry, 1 announced)
 BGP group iBGP type Internal
     Nexthop: 172.17.6.10
     Flags: Nexthop Change
     Localpref: 100
     AS path: [3320] I
     Communities: color:0:2222
```

SR TE POLICY – BGP EXAMPLE (1 OF 3)

- Controller advertises SR TE Policy via BGP

```
root@vmxdockerlight_pe1_1> show route receive-protocol bgp 172.22.63.254 detail
bgp.inetcolor.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
* 172.17.2.6-1111<sr>/96 (1 entry, 0 announced)
     Accepted
     SR Policy Distinguisher: 0
     SR Policy Color: 1111
     SR Policy Endpoint: 172.17.2.6
     Nexthop: 172.17.2.6
     Localpref: 200
     AS path: I
     Communities: no-advertise
                SRTE Policy Path:
                   Path preference: 100
                   Binding-SID: 1000013
                   Segment list:
                      Weight: 10
                      Label: 801002      ttl: Local-ttl-policy
                      Label: 807101      ttl: Local-ttl-policy
                      Label: 807102      ttl: Local-ttl-policy
                      Label: 807103      ttl: Local-ttl-policy
                      Label: 800006      ttl: Local-ttl-policy
                   <...>
```

SR TE POLICY – BGP EXAMPLE (2 OF 3)

- BGP installs received policies in new RIBs bgp.inetcolor.0/bgp.inet6color.0
- Selected policies installed in additional new RIBs inetcolor.0/inet6color.0 used to resolve steering routes
- Binding-SID label entry installed in mpls.0 table

```
root@vmxdockerlight_pe1_1> show route

mpls.0: 21 destinations, 21 routes (21 active, 0 holddown, 0 hidden)
1000013            *[BGP/170] 03:17:15, localpref 200, from 172.22.63.254
                      AS path: I, validation-state: unverified
                    > to 172.22.66.3 via ge-0/0/0.0, Swap 800006, Push 807103, Push 807102, Push 807101, Push 801002(top)
                      to 172.22.67.3 via ge-0/0/1.0, Swap 800006, Push 807103, Push 807102, Push 807101, Push 801002(top)

bgp.inetcolor.0: 1 destinations, 1 routes (1 active, 0 holddown, 0 hidden)
172.17.2.6-1111<sr>/96
                   *[BGP/170] 03:17:15, localpref 200, from 172.22.63.254
                      AS path: I, validation-state: unverified
                    > to 172.17.2.6

inetcolor.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)
172.17.2.6-1111<c>/64
                   *[BGP/170] 03:17:15, localpref 200, from 172.22.63.254
                      AS path: I, validation-state: unverified
                      to 172.22.66.3 via ge-0/0/0.0, Push 800006, Push 807103, Push 807102, Push 807101, Push 801002(top)
                    > to 172.22.67.3 via ge-0/0/1.0, Push 800006, Push 807103, Push 807102, Push 807101, Push 801002(top)
```

SR TE POLICY – BGP EXAMPLE (3 OF 3)

Steering routes ("colored") resolved via inetcolor.0/inet6color.0

```
root@vmxdockerlight_pe1_1> show route 11.0.1.11/32

inet.0: 33 destinations, 33 routes (33 active, 0 holddown, 0 hidden)
11.0.1.11/32       *[BGP/170] 00:14:19, localpref 100, from 172.17.2.6
                      AS path: I, validation-state: unverified
                    > to 172.22.66.3 via ge-0/0/0.0, Push 800006, Push 807103, Push 807102, Push 807101, Push 801002(top)
                      to 172.22.67.3 via ge-0/0/1.0, Push 800006, Push 807103, Push 807102, Push 807101, Push 801002(top)

root@vmxdockerlight_pe1_1> show route 11.0.1.11/32 detail

inet.0: 33 destinations, 33 routes (33 active, 0 holddown, 0 hidden)
11.0.1.11/32 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                <...>
                Next hop: 172.22.66.3 via ge-0/0/0.0 weight 0x1, selected
                Label operation: Push 801002
                <...>}
                Next hop: 172.22.67.3 via ge-0/0/1.0 weight 0xf000
                Label operation: Push 801002
                <...>
                Protocol next hop: 172.17.2.6-1111<c>
                <...>
                Communities: color:0:1111
```

SR TE POLICY – STATIC POLICY EXAMPLE (1 OF 3)

- Policy statically configured on ingress PE

```
protocols {
    source-packet-routing {
        segment-list P1-P3-PE2 {                          ← defines segment list
            P1 label 801001;                              ← first hop must be label value (not IP address)
            P3 label 801003;
            PE2 label 800006;
        }
        source-routing-path COLOR2-TO-PE2 {               ← static SR TE policy
            to 172.17.6.10;                               ← destination address of tunnel endpoint
            color 2222;                                   ← color identifier value
            binding-sid 1000222;                          ← Binding-SID label must be from static label range
            primary {
                P1-P3-PE2;                                ← references segment list
            }
        }
    }
}
```

SR TE POLICY – STATIC POLICY EXAMPLE (2 OF 3)

- Policy installed in inetcolor.0/inet6.color.0 to resolve steering routes
- Binding-SID label entry installed in mpls.0 table

```
root@vmxdockerlight_pe1_1> show route

mpls.0: 21 destinations, 21 routes (21 active, 0 holddown, 0 hidden)

1000222            *[SPRING-TE/8] 00:02:38, metric 1, metric2 20
                    > to 172.22.66.3 via ge-0/0/0.0, Swap 800006, Push 801003(top)

inetcolor.0: 2 destinations, 2 routes (2 active, 0 holddown, 0 hidden)

172.17.6.10-2222<c>/64
                   *[SPRING-TE/8] 00:02:38, metric 1, metric2 20
                    > to 172.22.66.3 via ge-0/0/0.0, Push 800006, Push 801003(top)
```

SR TE POLICY – STATIC POLICY EXAMPLE (3 OF 3)

- Steering routes ("colored") resolved via inetcolor.0/inet6color.0

```
root@vmxdockerlight_pe1_1> show route 11.0.1.22/32

inet.0: 33 destinations, 33 routes (33 active, 0 holddown, 0 hidden)
11.0.1.22/32       *[BGP/170] 00:18:50, localpref 100, from 172.17.2.6
                      AS path: I, validation-state: unverified
                    > to 172.22.66.3 via ge-0/0/0.0, Push 800006, Push 801003(top)

root@vmxdockerlight_pe1_1> show route 11.0.1.22/32 detail

inet.0: 33 destinations, 33 routes (33 active, 0 holddown, 0 hidden)
11.0.1.22/32 (1 entry, 1 announced)
        *BGP    Preference: 170/-101
                <...>
                Next hop: 172.22.66.3 via ge-0/0/0.0 weight 0x1, selected
                <...>
                Protocol next hop: 172.17.6.10-2222<c>
                <..>
                Communities: color:0:2222
```

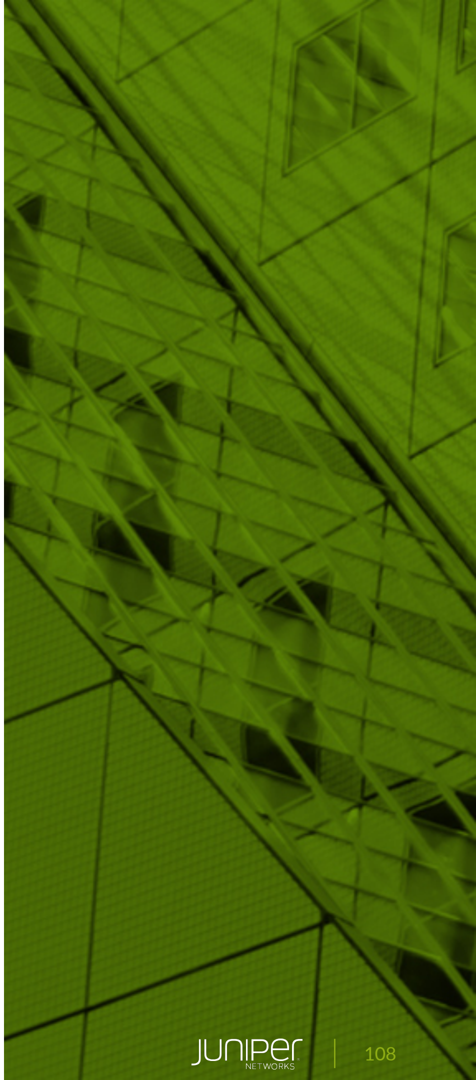
SR TE SEGMENT-LIST WITH AUTO-TRANSLATE

- Segment-List uses the Keyword “auto-translate“ to dynamically translates IP Address to SID

```
segment-list to_AG3-2 {
    inherit-label-nexthops;
    auto-translate;
    AG1-1 ip-address 221.2.0.25;
    AG2-1 ip-address 221.2.0.5;
    AG3-2 {
        ip-address 221.0.0.2;
        label-type {
            node;
        }
    }
}
source-routing-path to_AG3_2 {
    to 221.0.0.2;
    inactive: color 200;
    primary {
        to_AG3-2;
    }
}
```

FLEXIBLE ALGORITHM

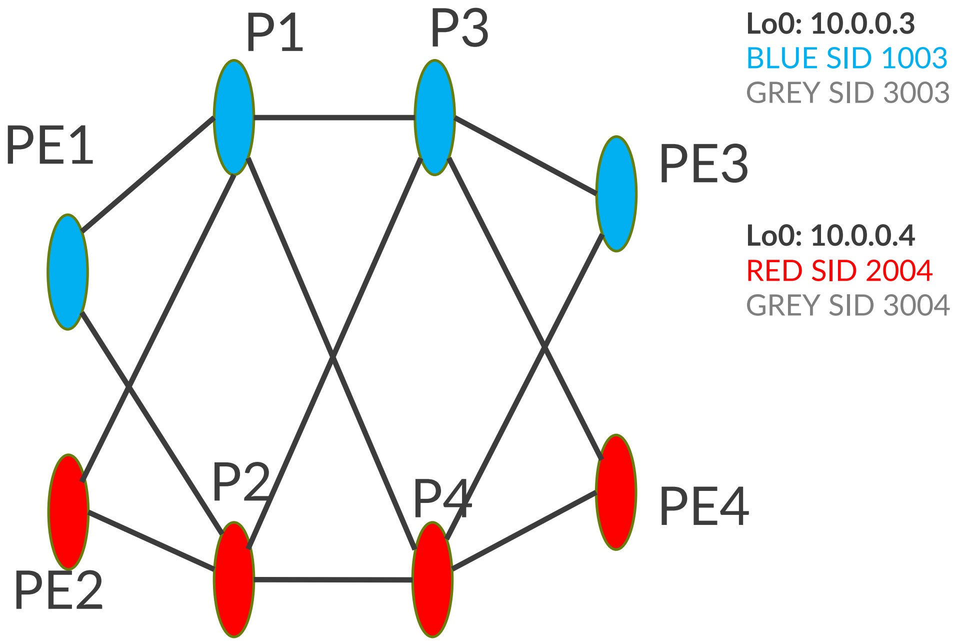
Simplified Traffic Engineering



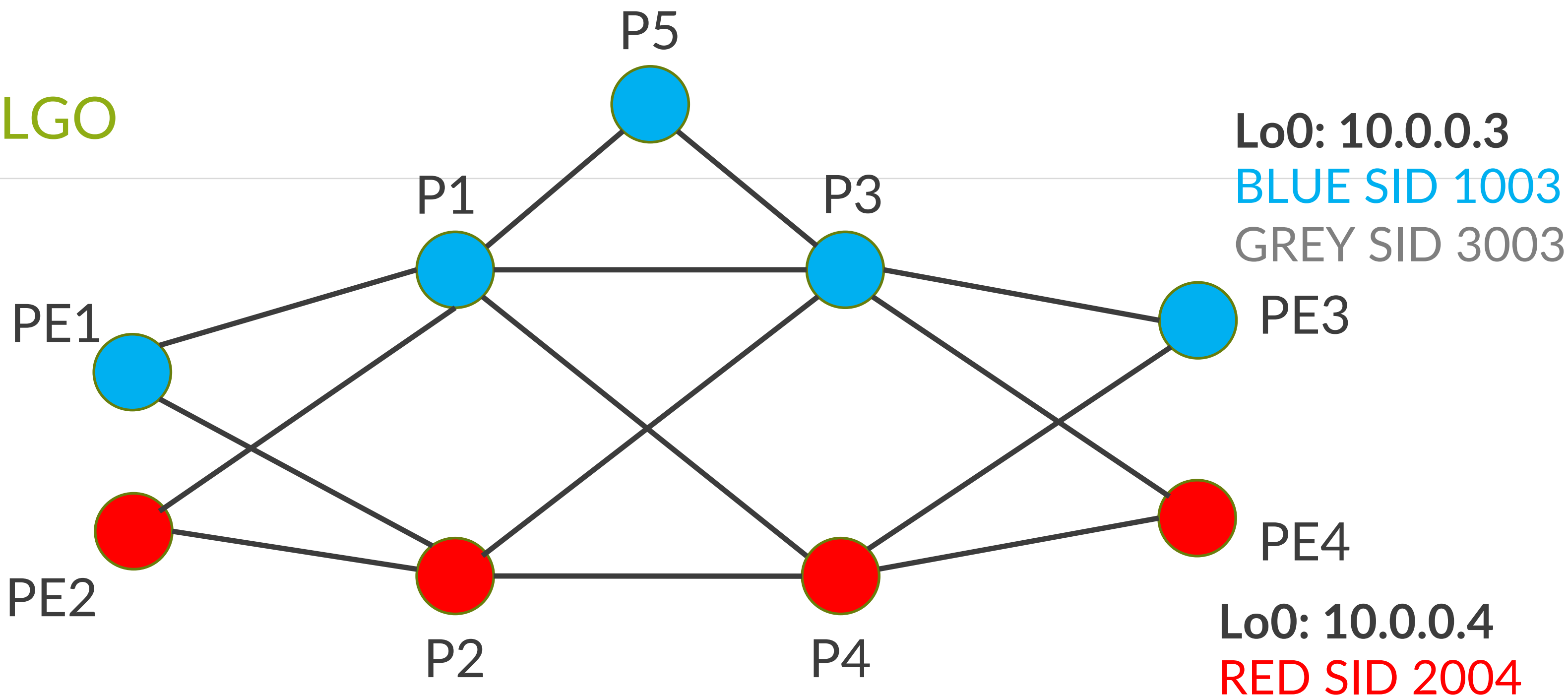
FLEX-ALGO INTRODUCTION (CONTD.)

- Mechanism to create paths for different Algorithms
- Algorithms could define
  - Different computation algorithms
  - Different metric-type ex: latency
  - Different constraints ex: link color

- Separate Routing-tables for each algorithm
- Mechanism to advertise separate SR-SIDs
- MPLS paths corresponding to the algorithm
- Sub-second FRR convergence with TI-LFA
- ECMP and Load-balancing by default
  - draft-gulkohegde-routing-planes-using-sr-00
  - Moved to draft-ietf-lsr-flex-algo now
  - EANTC Interpop

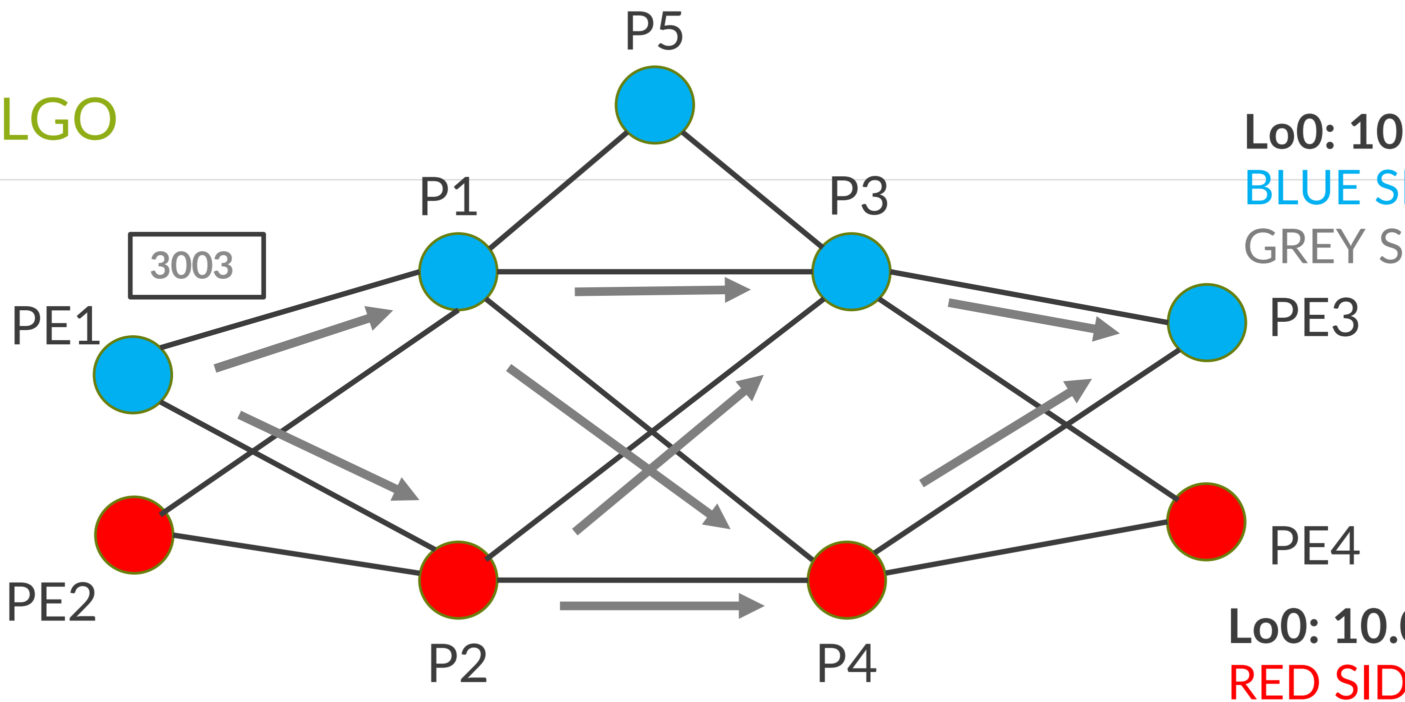


FLEX-ALGO



- A node can be a member of multiple algos.
- A Node announces a different node SID for each algo that it is a member of.
- Separate SPF per algo.
- In the diagram, all nodes are members of the Grey algo. Nodes in one plane are members of Red algo, nodes in the other plane are members of the Blue algo.
- A algo can be given a color, to allow auto-mapping of traffic to the correct algo.

FLEX-ALGO

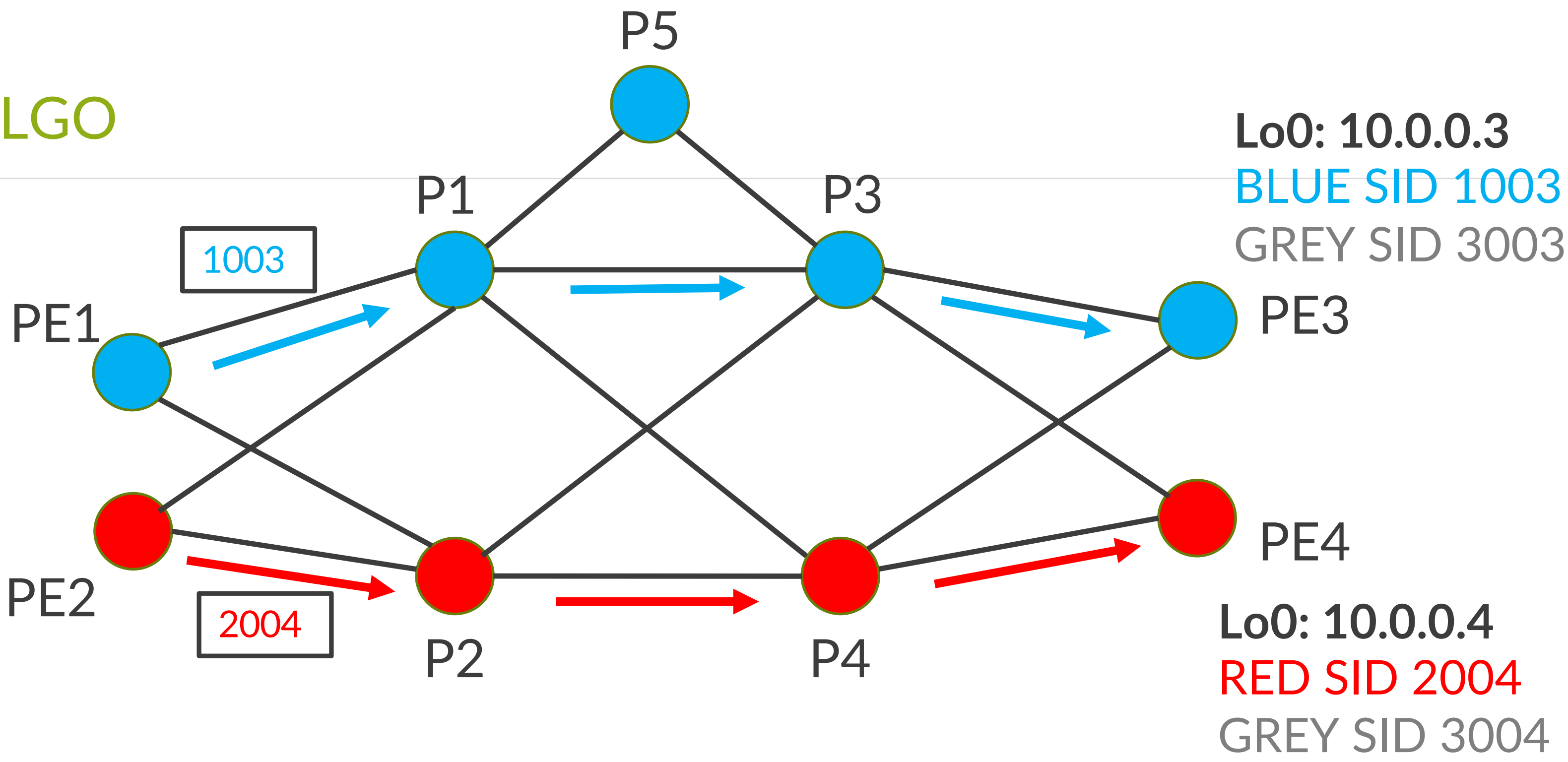


Lo0: 10.0.0.3
BLUE SID 1003
GREY SID 3003

Lo0: 10.0.0.4
RED SID 2004
GREY SID 3004

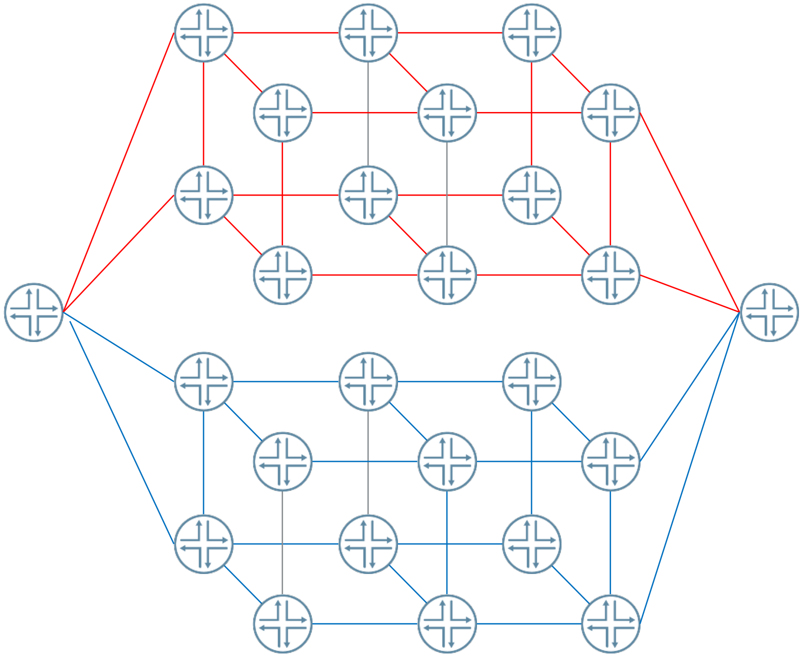
"Vanilla" traffic from PE1 is mapped to Grey algo, can go anywhere

FLEX-ALGO



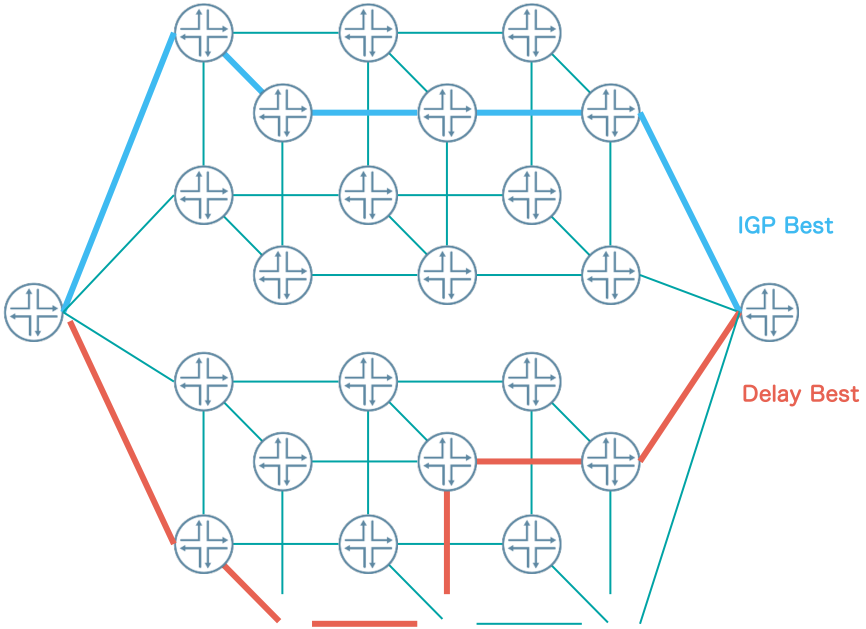
- We have some traffic from PE1 to PE3 that needs to be diversely routed from other traffic from PE2 to PE4
- Use Blue and Red algos respectively

Use-Case: Dual Plane Core



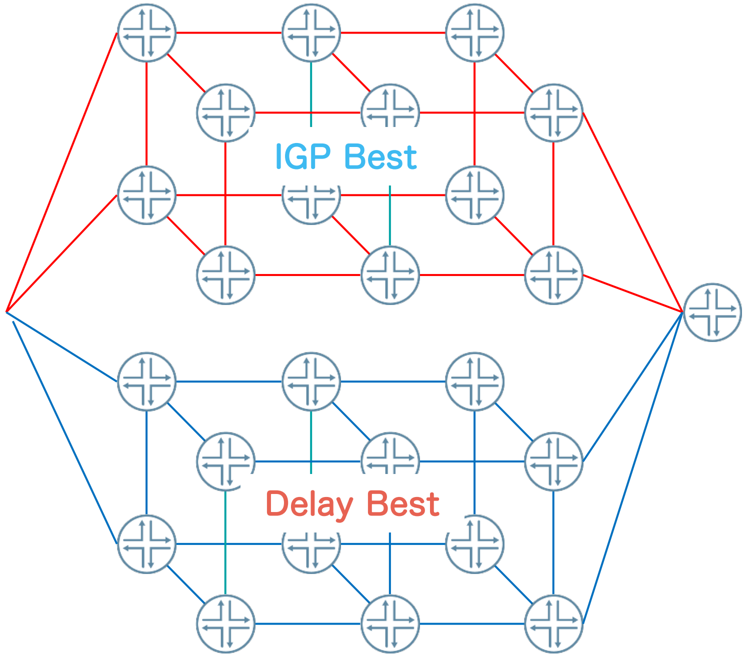
- Algo 128 for Core 1
  - ID 128, metric IGP, link Red
- Algo 129 for Core 2
  - ID 129, metric IGP, link Blue
- Service Traffic A = Core 1
- Service Traffic B = Core 2
- Calculation of SPF, advertising SIDs and TI-LFA are executed in each Core Plane separately
- Core Plane 1&2 is logically separate

Use Case: Low latency path



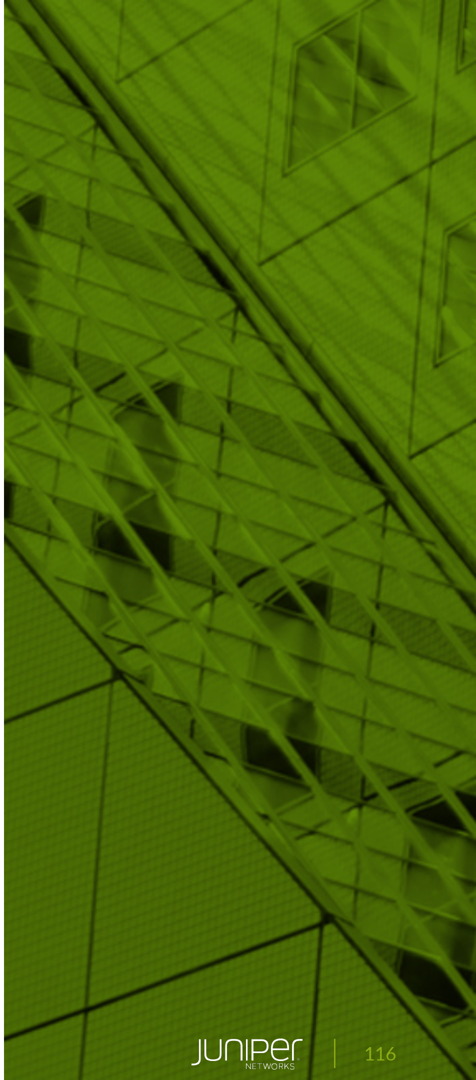
- Algo 128
  - ID 128, metric IGP
- Algo 129
  - ID 129, metric Delay
- Normal Service Traffic is dealt with Algo 128
- Delay sensitive Service Traffic is dealt with Algo 129
- All of links in domain are usable

Use Case: Dual Plane + Different Metric



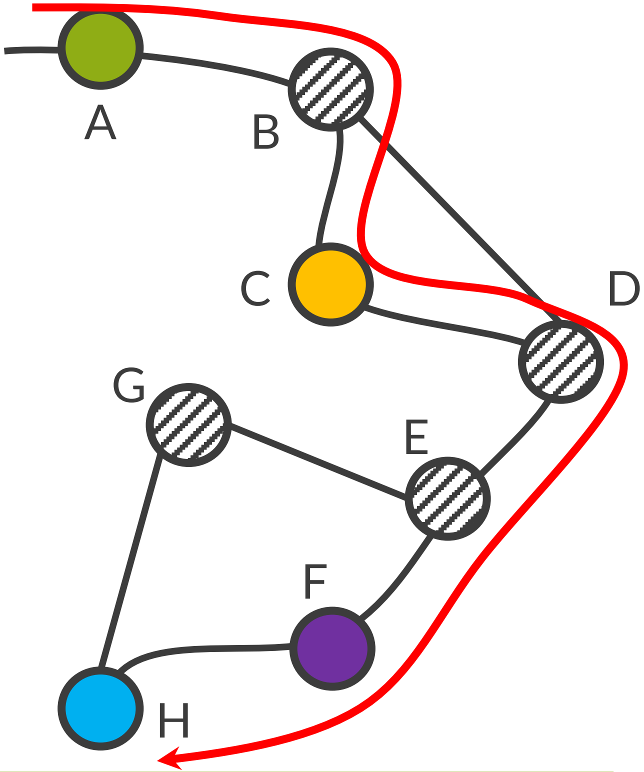
- Algo 128 for Core 1
  - ID 128, metric IGP, link Red
- Algo 129 for Core 2
  - ID 129, metric Delay, link Blue
- Normal Service Traffic is dealt with at Core 1 (IGP Best domain)
- Delay sensitive Service Traffic is dealt with at Core 2 (Delay Best)
- Calculation of SPF, advertising SIDs and TI-LFA are executed each Core Plane separately
- Core Plane 1&2 is logically separated

SRV6 AND SRV6+

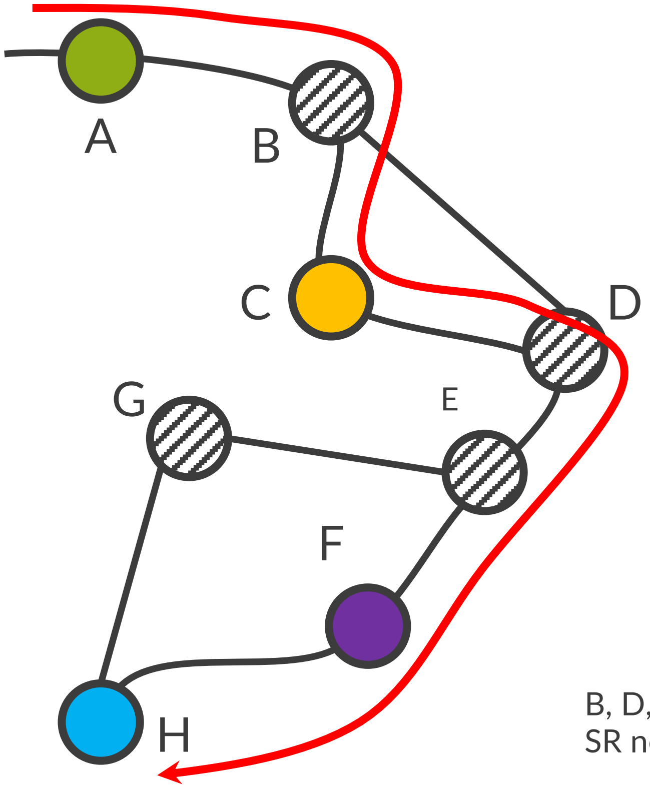


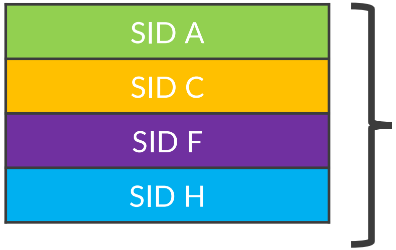
SRv6 INTRODUCTION

- Uses IPv6 address instead of MPLS labels as SIDs
- Can be used for **traffic engineering** and to carry instructions for VNFs in NFV deployments
- Unlike SR-MPLS, SRv6 SIDs are not popped –the entire stack of SIDs stays with the packet.
    - This means the destination knows which source (and intermediate segments) the packet has come from.



SRv6 CONCEPT





Stack of SIDs within IPv6 header.

Each SID is a 128-bit IPv6 "address".

- A inserts SRH with SID list as set of IPv6 IP addresses
- B Transient/Ordinary node process IPv6 header
- C Segment Endpoints copy appropriate SIDs (F) from SRH in DA of IPv6 Header and decrement Segment Left
- F Segment Endpoint copy appropriate SID (H) from SRH in DA of IPv6 Header and decrement Segment Left
- H Strip SRH and forward IPv6 packet
- Segment Endpoint process IPv6

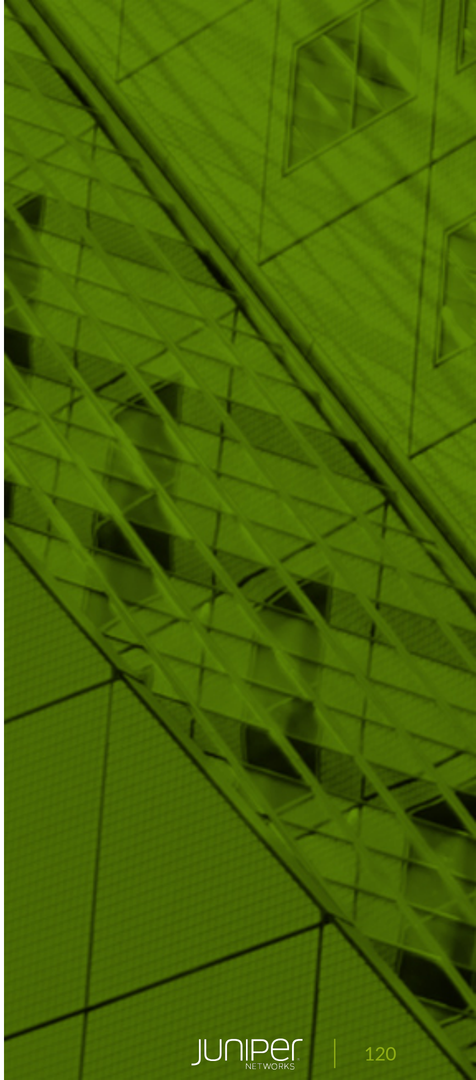
B, D, E and G are "ordinary" nodes, not segment endpoints.
SR nodes C and H perform VNF functions, SR node F does not.

SRv6 – SRH SEGMENT ROUTING HEADER

CURRENT STATE

- Each SID in the stack is represented by a 16-byte or 128 bit IPv6 address
- Plus the 8 bytes at the start of the Segment Routing Header (SRH)
- Usually 8 SIDs in header = 136 bytes of overhead
- Average internet packet size of 576 bytes becomes 712 bytes, which means SRv6 overhead uses **19 Gbps of a 100 Gbps link!**

SRv6+ OR SIMPLIFIED SRv6



TWO SID CLASSES

TRANSPORT SIDS

- Steers packets to the terminal segment
- Processed at non-terminal segment endpoints (SL > 0)
- Example: END, END.X
- Relatively few of these
- Simple semantic
  - Carry relatively little information

SERVICE SIDS

- Determines behavior at the terminal segment
- Processed at terminal segment endpoint only (SL = 0)
- Example: END.DX4, END.DX6
- Relatively many of these
- Rich semantic
  - Carry more information

IPV6 - TWO WAYS TO DELIVER INSTRUCTIONS TO DOWNSTREAM NODES

ROUTING EXTENSION HEADER

- Steers packets from ingress to egress
- Processed at non-terminal segment endpoints (SL > 0)
- Well-positioned to carry Transport SIDs

DESTINATION OPTIONS HEADER

- Determines behavior at egress node
- Processed at terminal segment endpoint only (SL = 0)
- Well-positioned to carry Service SIDs

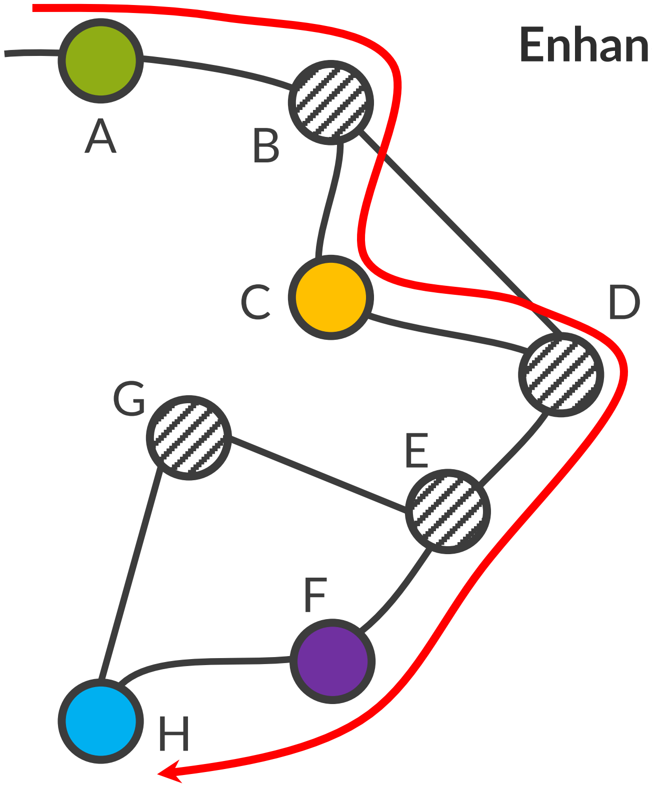
SRv6+ CRH COMPRESSED ROUTING HEADER

Enhancement

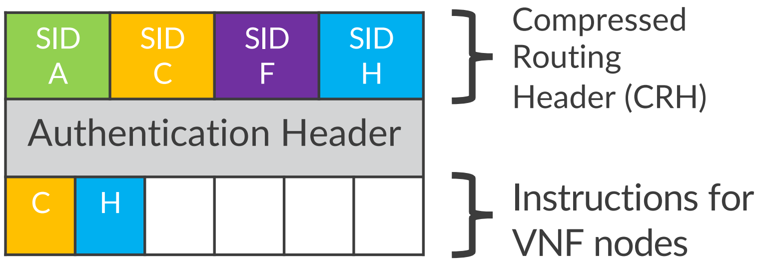
- Instead of 16-byte (128 bit) SID, use shorter SIDs.
- 8-bit, 16-bit or 32-bit SIDs, depending on the network. Distributed in same way as vanilla SIDs (e.g. via IGP)
- Example 8 SIDs in header with 4 byte SIDs = 40 bytes of overhead
- Average internet packet size of 576 bytes becomes 616 bytes, which means SRv6+ uses only **6 Gbps of a 100 Gbps link!**

draft-bonica-6man-comp-rtg-hdr

SRv6+ DESTINATION OPTIONS



Enhancement



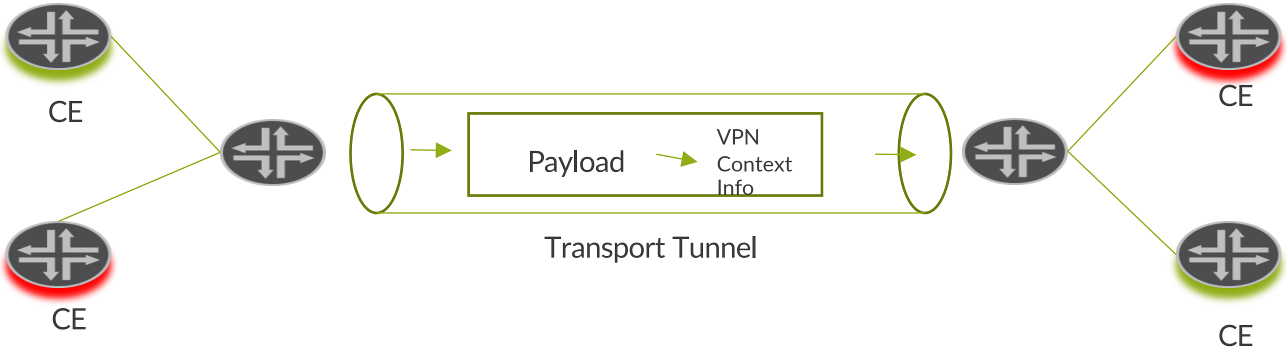
SRv6+ separates instructions from the SIDs, using *Segment Endpoint Option*

- Avoids overloading IPv6 address semantics
- More efficient. Not all segment endpoints perform processing, so they don't need an instruction
- Allows instructions to be authenticated, using an Authentication Header.

draft-bonica-6man-seg-end-opt/

SRv6+ VPN CONTEXT INFORMATION OPTION

- How do we carry the equivalent of the "VPN label" that exists in MPLS?
- SRv6 uses a SID
- In SRv6+, instead use new *VPN Context Information Option*
  - Can be authenticated, by using Authentication Header. Very important in an over-the-top environment, so we know the VPN info has not been tampered with en-route
  - Less bandwidth overhead



IPv6 OAM OPTION

Allows OAM to be triggered on:

- Destination node only *or*
- All nodes along the packet's journey *or*
- Only the nodes listed in the Routing Header (i.e. SRv6 nodes)

OAM actions that can be requested:

- Log, with timestamp
- Count
- Send telemetry, with timestamp
- Send ICMP message, with timestamp

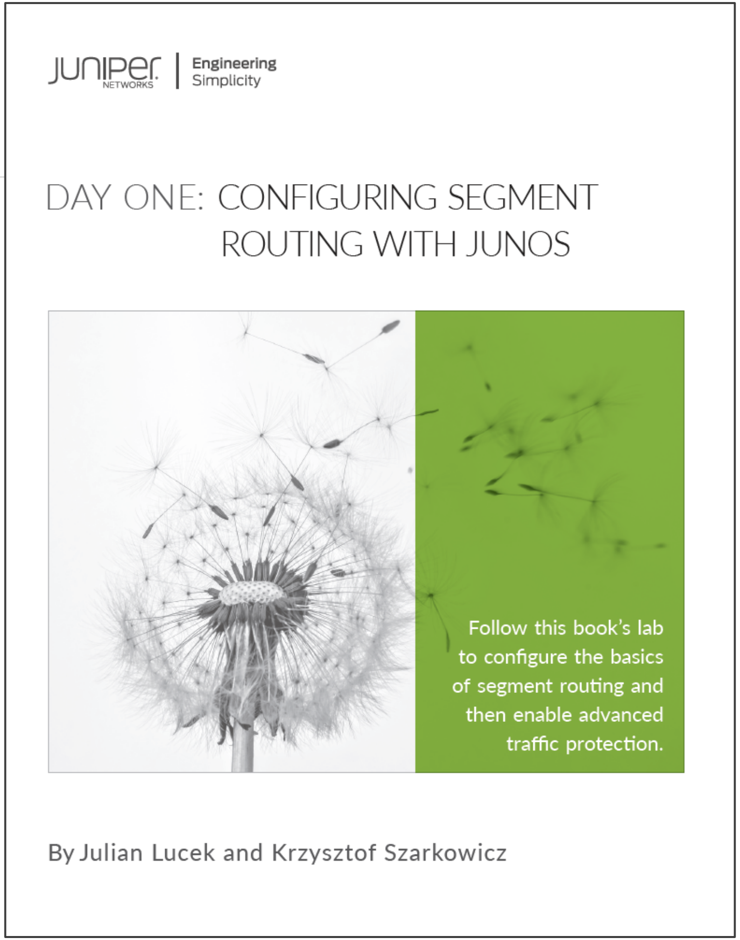
Step by Step Configuration of SPRING with JUNOS

Adj-SIDs, Node-SIDs, anycast SIDs, Adj-Sets

LDP-SR interworking

TI-LFA

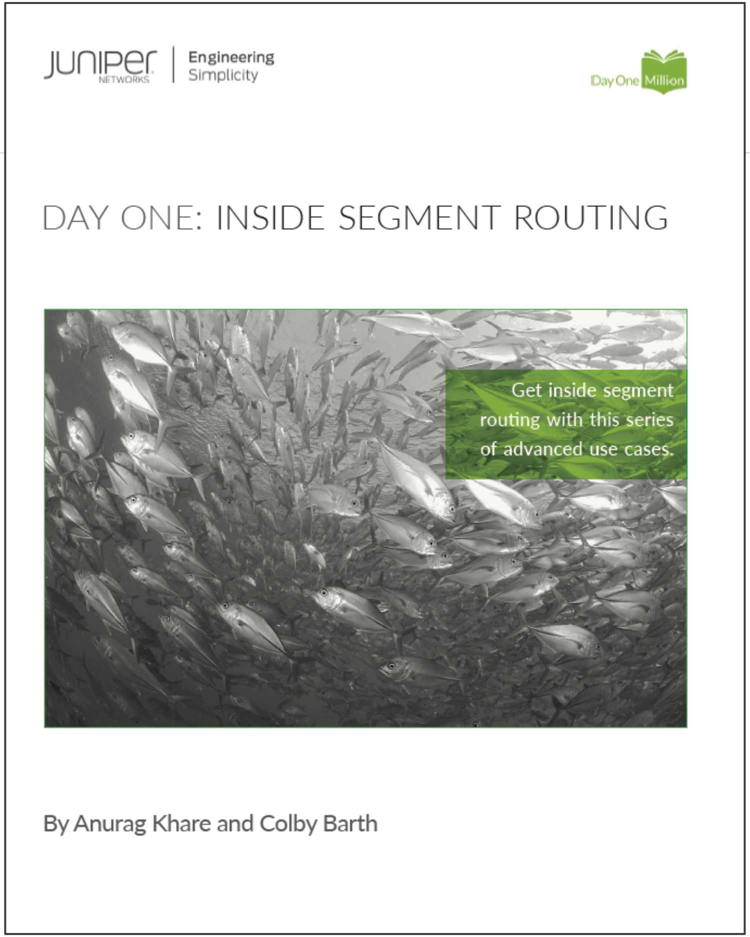
https://www.juniper.net/uk/en/training/jnbooks/day-one/configuring-segment-routing-junos/index.page



SPRING Topics covered include:

- Traffic Engineering
- NorthStar Controller
- SR over UDP

https://www.juniper.net/us/en/training/jnbooks/day-one/inside-segment-routing/



IETF DRAFTS

draft-ietf-spring-segment-routing
Segment Routing architecture

draft-ietf-spring-segment-routing-mpls
Segment Routing details with MPLS forwarding

draft-ietf-isis-segment-routing-extensions
ISIS extensions to distribute SR segments

draft-ietf-isis-prefix-attributes
Advertising ISIS prefix attributes for SR

draft-ietf-ospf-segment-routing-extensions
OSPF extensions to distribute SR segments

RFC 7684
OSPFv2 Prefix/Link Attribute Advertisement

draft-hegde-rtgwg-microloop-avoidance-using-spring-01
Micro-loop avoidance using SPRING

draft-francois-rtgwg-segment-routing-ti-lfa
FRR using TI-LFA.

IETF DRAFTS

draft-rosen-idr-rfc3107bis-00

Advertise a label stack using BGP-LU

draft-gredler-idr-bgp-ls-segment-routing-ext

BGP LS extensions for exporting SR topology to a controller (north bound)

draft-ietf-pce-segment-routing

PCE extensions to setup a SR path different from shortest path (SR-TE) from the controller (south bound)

draft-sreekantiah-idr-segment-routing-te

Advertise SR-TE policies via BGP

draft-sivabalan-pce-binding-label-sid

Advertise binding label SID to steer traffic through a TE LSP

draft-tantsura-bgp-ls-segment-routing-msd-00

draft-tantsura-ospf-segment-routing-msd-00

draft-tantsura-isis-segment-routing-msd-00

Exposing Max Label stack depth supported by a node

draft-hegde-spring-node-protection-for-sr-te-paths-00

Node Protection for SR-TE Paths

LIST OF IETF DRAFTS ABOUT SRv6+

https://datatracker.ietf.org/doc/draft-bonica-6man-seg-end-opt/

https://datatracker.ietf.org/doc/draft-bonica-6man-oam/

https://datatracker.ietf.org/doc/draft-bonica-6man-vpn-dest-opt/

https://datatracker.ietf.org/doc/draft-bonica-6man-comp-rtg-hdr/

THANK YOU

JUNIPER NETWORKS | Engineering Simplicity

Aditya Kaul
Solution Architect,
Professional Services-APAC
mobile +65 9622 5560
email kaula@juniper.net
twitter @KaulAddie

www.juniper.net

JUNIPER NETWORKS