SRv6 introduction

Paresh Khatri/Thomas Corre

26 May 2025

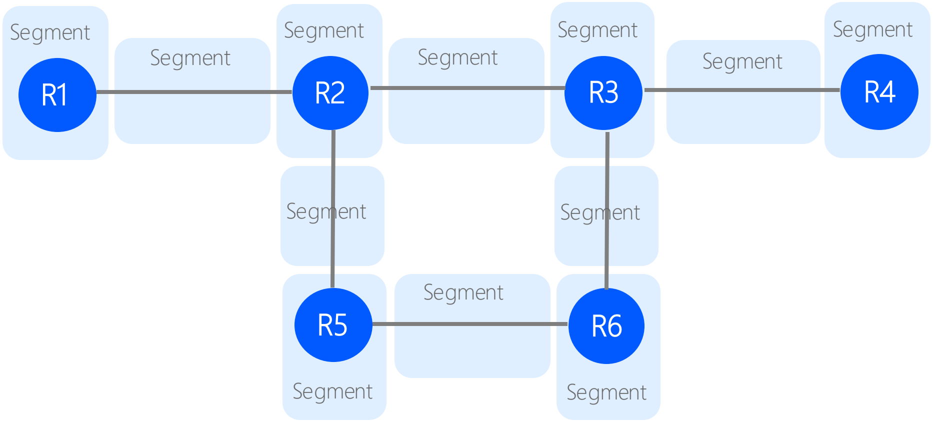
Agenda

1. Segment routing introduction
2. Basics of SRv6
3. Agenda item
4. Agenda item

Segment routing introduction

Segment routing introduction

- Segment Routing (SR) provides a tunneling mechanism via source routing
- A segment can represent a node, link, or service
- Topological segments are advertised by routing protocols (IS-IS and OSPF)
- A SR tunnel can be encoded as
  - A single MPLS label or a stack of MPLS labels
  - A single IPv6 address or several IPv6 addresses in the IPv6 Extension header (Segment Routing Header).
- The segments can be thought of as a set of instructions executed from the ingress PE



  Public

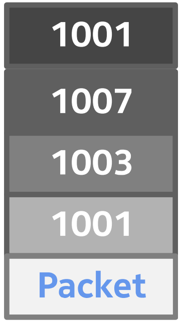
Encoding Segment Routing tunnels

A Segment Routing (SR) tunnel, containing a single segment or a segment list, is encoded as:

- A single MPLS label or an ordered list of hops represented by a stack of MPLS labels (no change to the MPLS data-plane).
- A single IPv6 address, or an ordered list of hops represented by a number of IPv6 addresses in the IPv6 Extension header (Segment Routing Header).

The segment list can represent either a topological path (node, link) or a service.

The segments can be thought as a set of instructions from the ingress PE such as “go to node D using the shortest path”, “go to node D using link/node/explicit-route L”



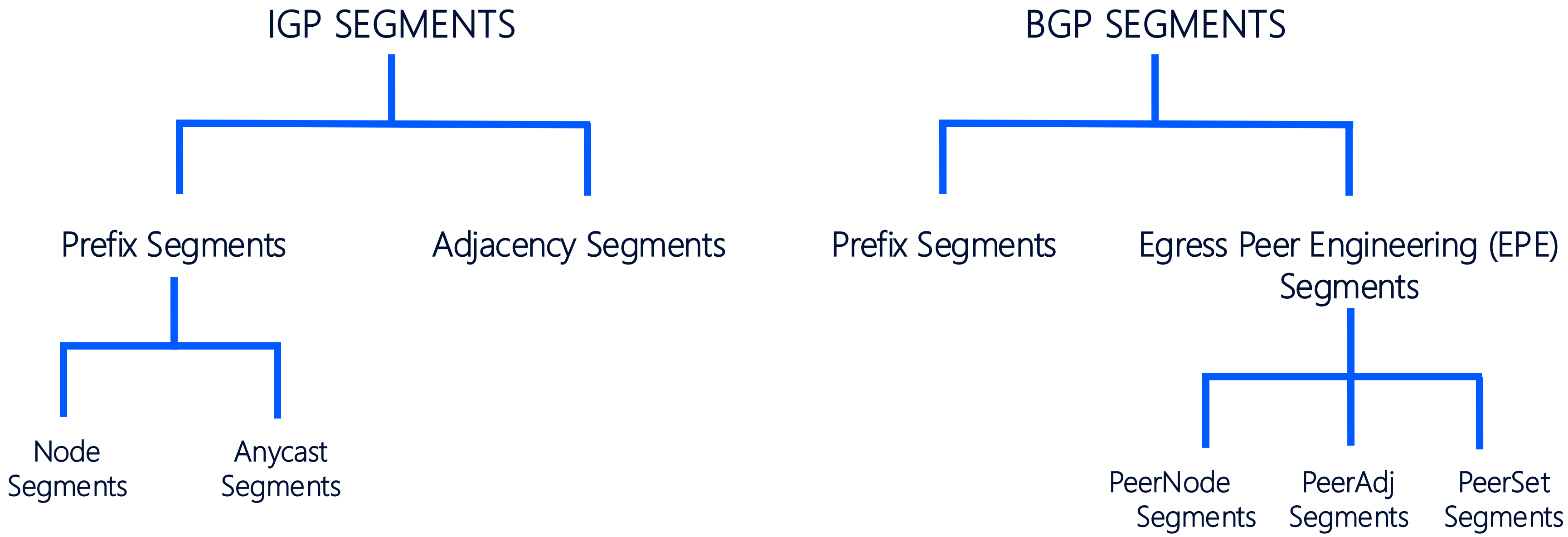
Segment routing with MPLS data plane

- MPLS instantiation of Segment Routing aligns with the MPLS architecture defined in RFC 3031
- For each segment, the IGP advertises an identifier referred to as a Segment ID (SID). A SID is a 32-bit entity; with the MPLS label being encoded as the 20 right-most bits of the segment ID.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Label
|                Label                  | TC  |S|       TTL     | Stack
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Entry

                    Label:  Label Value, 20 bits
                    TC:     Traffic Class field, 3 bits
                    S:      Bottom of Stack, 1 bit
                    TTL:    Time to Live, 8 bits
```

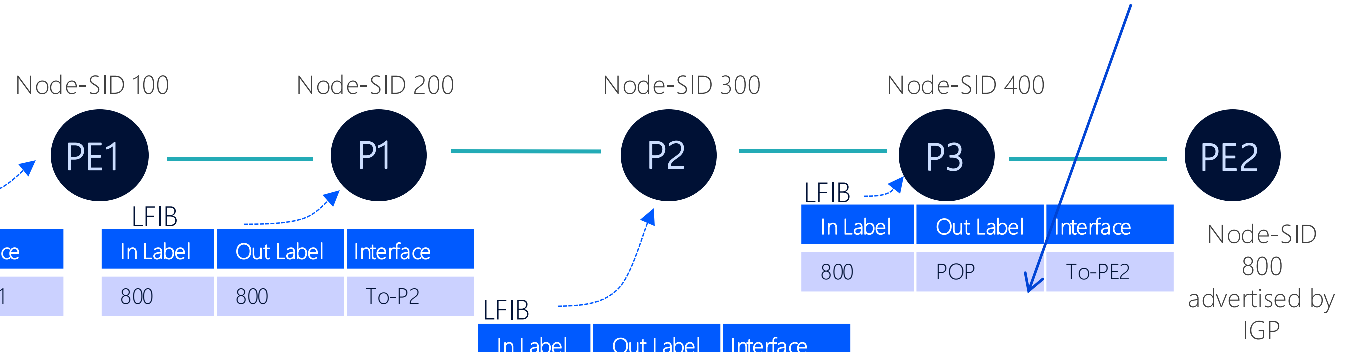
Types of segments

Taxonomy



Example: SR tunnel with prefix-SID (node-SID) [1]

- PE2 advertises Node Segment into IGP (Prefix-SID Sub-TLV Extension to IS-IS/OSPF)
- All routers in SR domain install the node segment to PE2 in the MPLS data-plane.
  - No RSVP and/or LDP control plane required.
  - When applied to MPLS, a Segment is essentially an LSP.



FEC to NHLFE

FEC	Out Label	Interface
PE2	800	To-P1

LFIB

In Label	Out Label	Interface
800	800	To-P2

LFIB

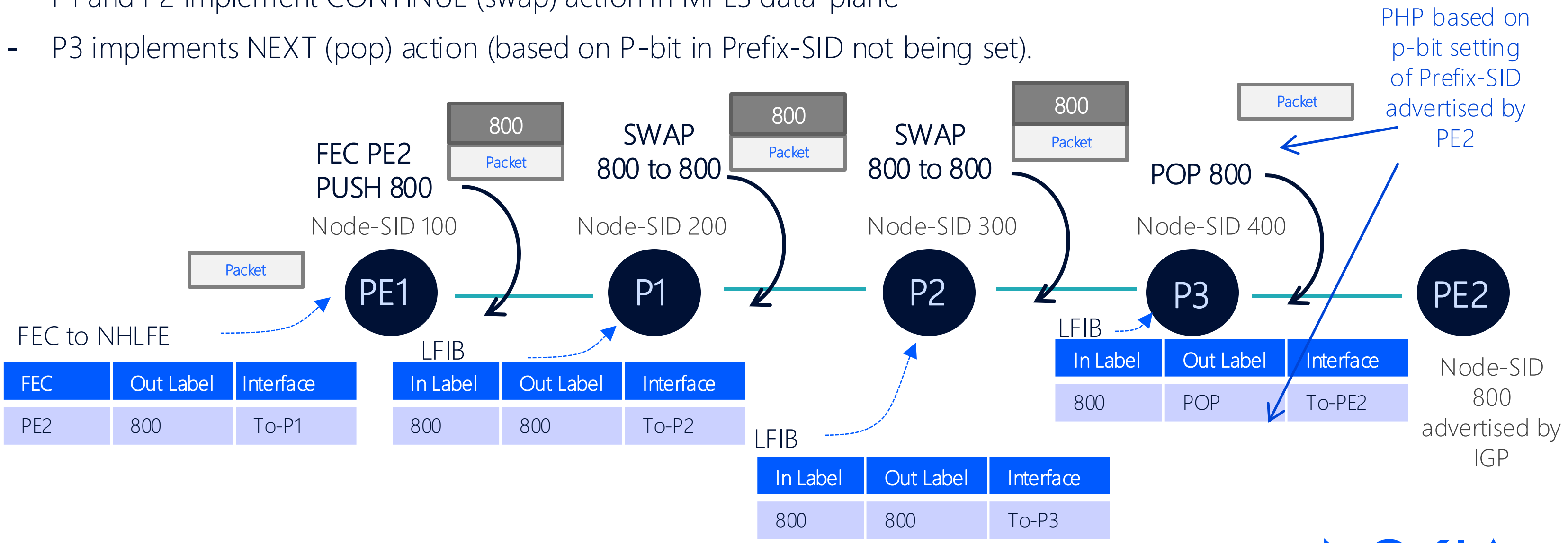
In Label	Out Label	Interface
800	800	To-P3

LFIB

In Label	Out Label	Interface
800	POP	To-PE2

Example: SR tunnel with prefix-SID (node-SID) [2]

- For traffic from PE1 to PE2, PE1 pushes on node segment {800} and uses shortest IGP path to reach PE2.
- Active segment is the top of the stack for MPLS:
  - P1 and P2 implement CONTINUE (swap) action in MPLS data-plane
  - P3 implements NEXT (pop) action (based on P-bit in Prefix-SID not being set).



FEC to NHLFE (PE1)

FEC	Out Label	Interface
PE2	800	To-P1

LFIB (P1)

In Label	Out Label	Interface
800	800	To-P2

LFIB (P2)

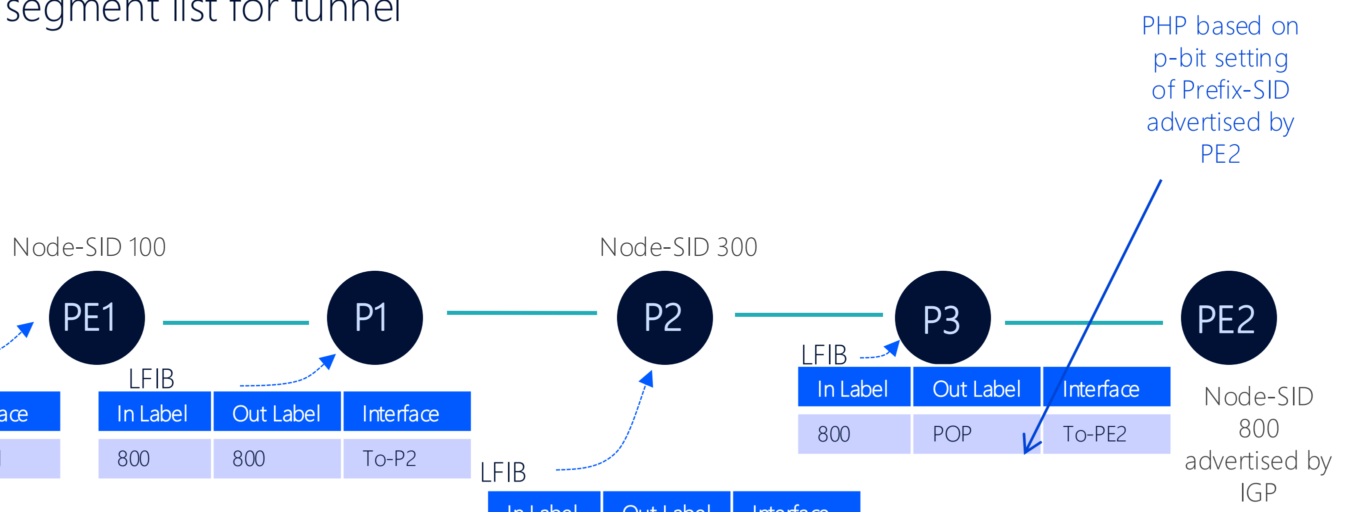
In Label	Out Label	Interface
800	800	To-P3

LFIB (P3)

In Label	Out Label	Interface
800	POP	To-PE2

Example: SR tunnel with prefix-SID (node-SID) [3]

- No per-path state held in network with the exception of segment list for tunnel held at PE1.



FEC to NHLFE

FEC	Out Label	Interface
PE2	800	To-P1

LFIB

In Label	Out Label	Interface
800	800	To-P2

LFIB

In Label	Out Label	Interface
800	800	To-P3

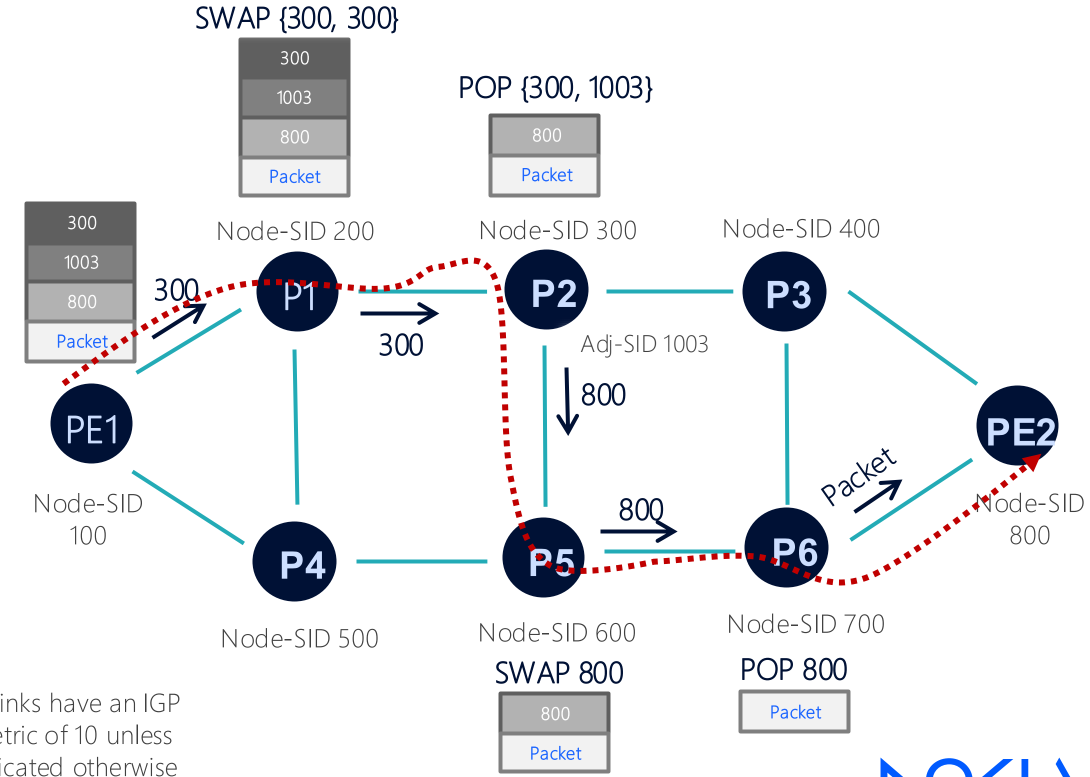
LFIB

In Label	Out Label	Interface
800	POP	To-PE2

Public

Example: SR tunnel with node and adjacency segments

- In this example, PE1 wants to traverse the link P2-P5 on the way to PE2, as it is under-utilised.
- PE1 therefore imposes the segment list {300, 1003, 800} representing the Node-SID for P2, the Adj-SID for link P2-P5, and finally the Node-SID for PE2.

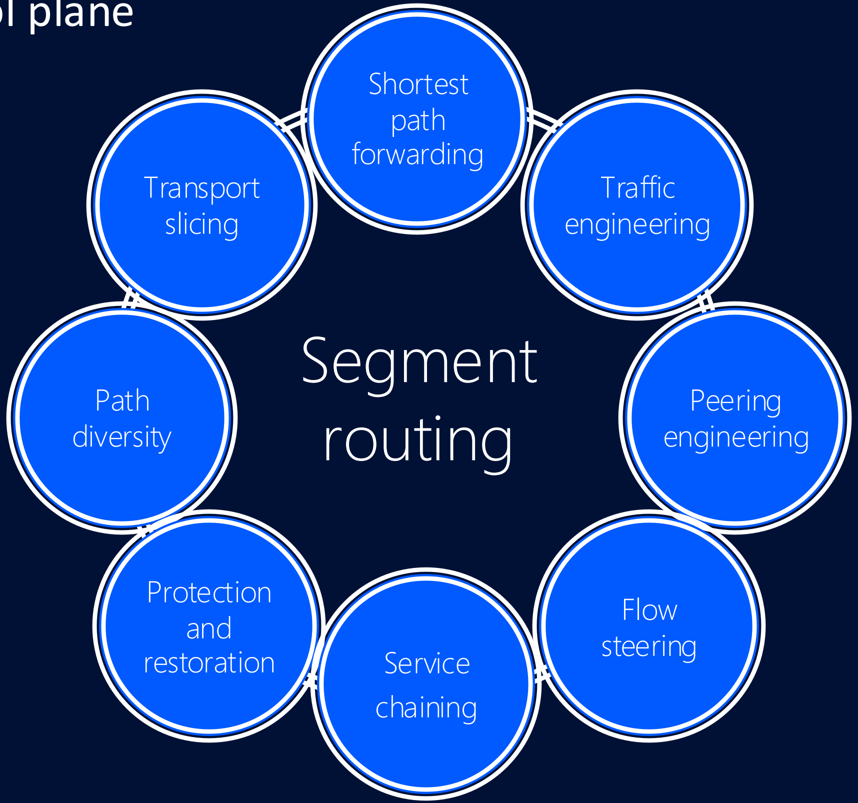


All links have an IGP metric of 10 unless indicated otherwise

Segment routing benefits

Versatile capabilities with a simple control plane

- Reduce control plane protocols
- Improve Scale / Optimize State
- Ease IPv6 Migration
- Enhance programmability & SDN Control



  Public

Basics of SRv6

SRv6 Drivers and Choices

Main drivers at a glance

More "Purist" approach to Segment Routing

No more MPLS protocols and labels

More Friendly to "Network Programming"

SIDs with structures and semantics

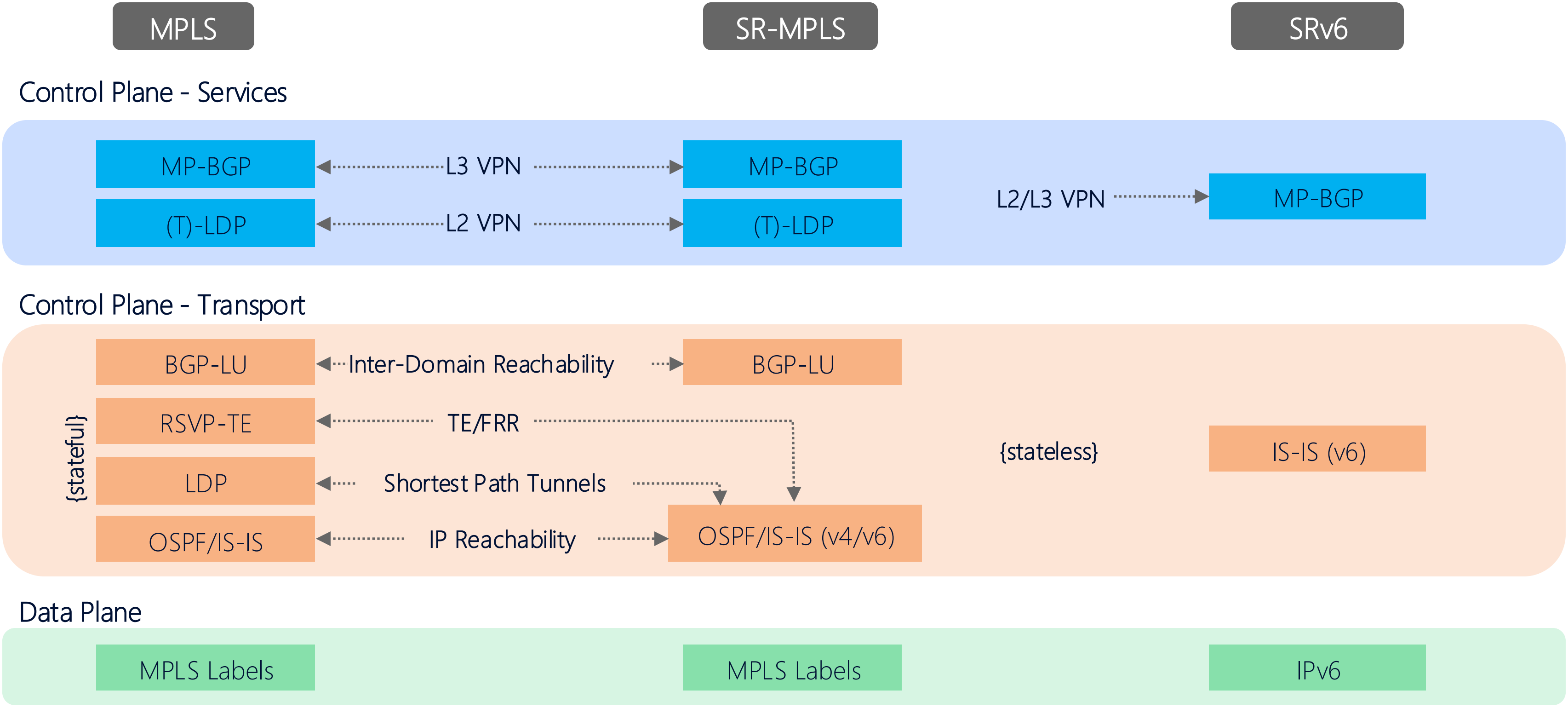
Easier to incorporate non-MPLS nodes (Servers, VMs, Containers)

Easier to incorporate non-SRv6-capable nodes

  Public

SRv6 Drivers and Choices

Control Plane Simplification



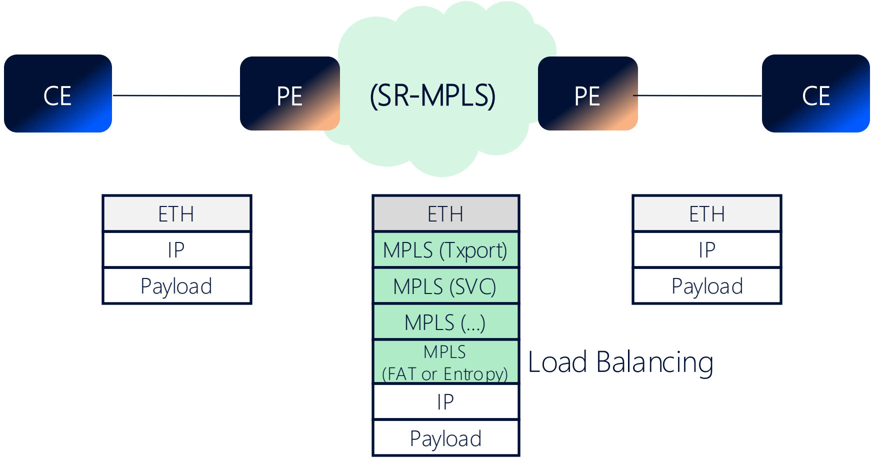
  Public

SRv6 Drivers and Choices

Data Plane Simplification

MPLS Header

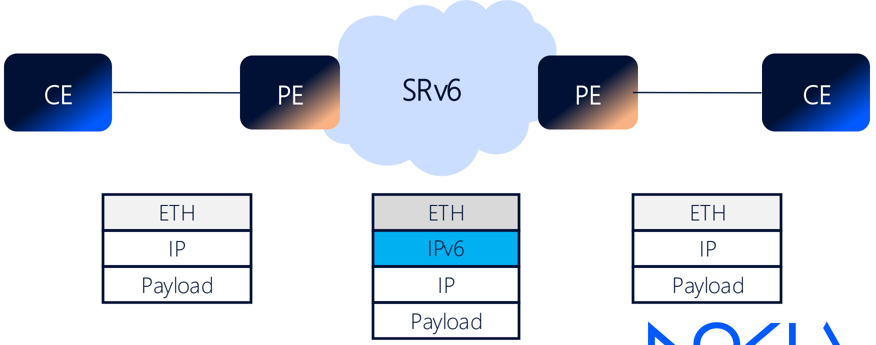
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Label
|                Label                  | Exp |S|       TTL     | Stack
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Entry

                    Label:  Label Value, 20 bits
                    Exp:    Experimental Use, 3 bits
                    S:      Bottom of Stack, 1 bit
                    TTL:    Time to Live, 8 bits
```



IPv6 Header

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  | Load Balancing
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

128 bits (16 Bytes)



IPv6 capabilities inherited by SRv6

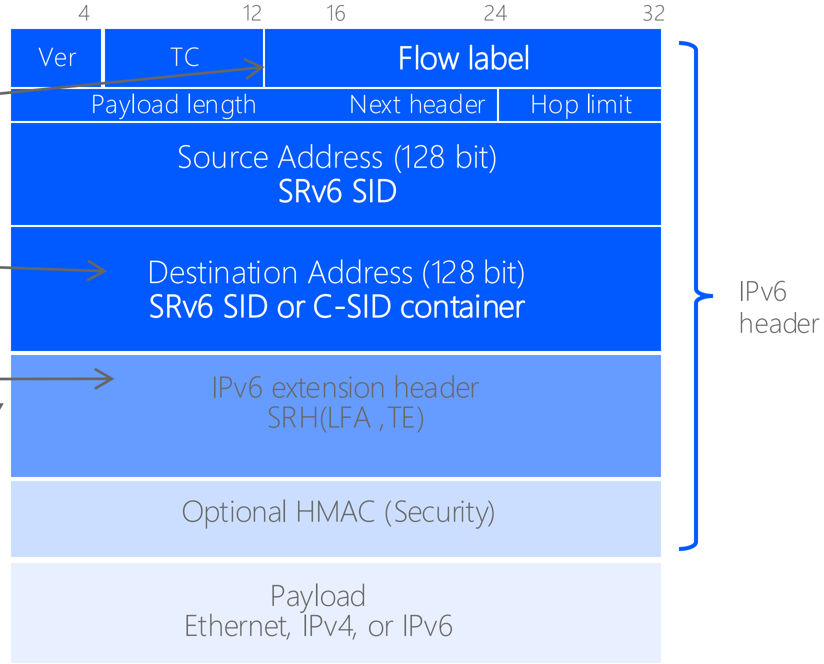
Native entropy for load balancing

Used in shortest path routing, SRv6 SID = IPv6 DA

Flexible header may encode instructions to next hop, to the destination, or multiple hops in between.

Segment Routing Header (SRH) is type 4 routing header

SRv6 also inherits ICMPv6 tools



Flexibility and programmability

Basics of SRv6

SRv6 SID

Segment routing identifier

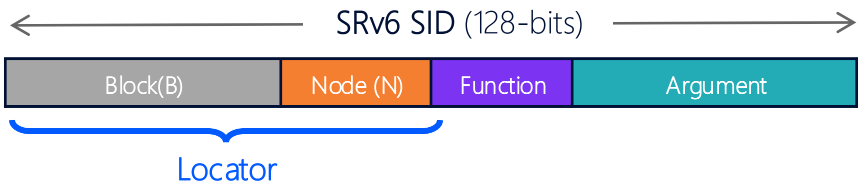
SRv6 SID components

- SR is applied to the IPv6 data plane using 128-bit SIDs that consist of a Locator, Function and an Argument.

- The Locator is broken into a Block and a Node:
  - The **Block** is the prefix allocated to SRv6 by the operator.
  - The Node identifies the parent node instantiating the SID.
  - The Locator is flooded into the IGP using an SRv6 Locator TLV and leads to the node instantiating the SID.

An SRv6 SID is a 128-bit IPv6 Address associated with a segment.

- An SRv6 SID is a routable IPv6 prefix when it is set as the IPv6 Header destination address.
  - Note: IPv6 router addresses (e.g. system address, router interface address) are *not* IPv6 SIDs.

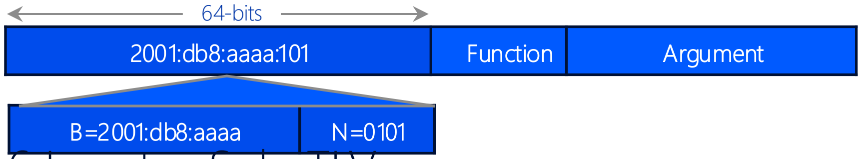


SRv6 Address Blocks – Locator

SRv6 SID structure and Locator visibility

- The Locator part of a SID is of the format B:N::
- Example: allocate a /64 locator for a set of routers in a routing domain.
  - Starting with a pre-allocated 32-bit SID address block. - 2001:db8::/32.
  - Allocate a /48 to all routers in an SRv6 routing domain – 2001:db8:aaaa::/48 (This is the common block B)
  - Each router in the domain has a /16 node identifier allocated - :0000 to :ffff. (node block N)
  - Examples: Locator prefix for router
    - PE1 = 2001:db8:aaaa:101::/64
    - PE2 = 2001:db8:aaaa:102::/64
    - PE3 = 2001:db8:aaaa:103::/64

SRv6 SID Locator Encoding example



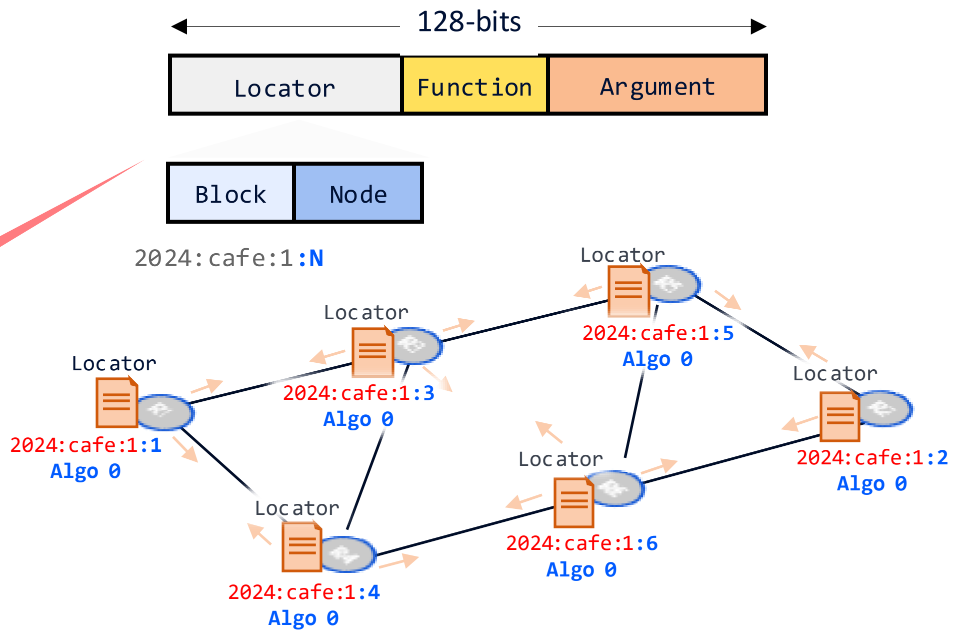
- Local router installs locator in IPv6
- The Locator prefix is advertised in ISIS in SRv6 Locator Sub-TLV.
  - Each remote router populates RTM and regular Fib with Locator prefix.
    - Locator Prefix is subject to IPv6 prefix match, with tunnelled next-hop to originating router.

  Public

SID Format and Locator Advertisement

SRv6 SID structure and Locator visibility

```
⌄ ISO 10589 ISIS Link State Protocol Data Unit
    PDU length: 391
    Remaining lifetime: 1200
    LSP-ID: 1920.0000.2001.00-00
    Sequence number: 0x0000009c
    Checksum: 0xd20b [correct]
    [Checksum Status: Good]
  > Type block(0x03): Partition Repair:0, Attached bits:0, Overload bit:0, IS type:3
  > Area address(es) (t=1, l=4)
  > Protocols supported (t=129, l=2)
  > Hostname (t=137, l=2)
  > Traffic Engineering Router ID (t=134, l=4)
  > Router Capability (t=242, l=60)
  > IP Interface address(es) (t=132, l=12)
  > IPv6 Interface address(es) (t=232, l=16)
  > Extended IS reachability (t=22, l=52)
  > Extended IS reachability (t=22, l=52)
  > Extended IP Reachability (t=135, l=44)
  > IPv6 reachability (t=236, l=36)
  ⌄ SRv6 Locator (t=27, l=56)
      Type: 27
      Length: 56
      0000 .... .... .... = Reserved: 0x0
      .... 0000 0000 0000 = Topology ID: Standard topol
    ⌄ SRv6 Locator: 2024:cafe:1:1::/64 (Algorithm: 0)
        Metric: 1
      > Flags: 0x00
        Algorithm: Shortest Path First (SPF) (0)
        Locator Size: 64
        Locator: 2024:cafe:1:1::
        SubCLV Length: 22
      > subTLV: SRv6 End SID (c=5, l=20)
    > SRv6 Locator: 2024:cafe:129:1::/64 (Algorithm: 129)
```



NOKIA

Segment Routing over IPv6

- Segment Routing Traffic Engineering instantiated on IPv6 data plane (SRv6)
- SRv6 Segment-List - A sequence of instructions associated with a segment and each segment is encoded as a 128-bit IPv6 Address in the packet header
- A Segment List can be:
  - a single IPv6 Address encoded as the destination address in the IPv6 header (see table 1 for encoding)
  - The SID encoded as an IPv6 Address provides a tunnel for the IP header to an IPv6 destination.

Table 1 – SRv6 Shortest Path Routing

Header Type	Parameter encoding.
IPv6 Header	Next header = IP SA = 2001:db8:1::1 DA= 2001:db8:aaaa:101:0:1000
IP Header	Version 4, IHL=20 SA= 10.1.2.1, DA = 10.3.2.1 Protocol = UDP etc.

Segment Routing over IPv6

- Segment Routing Traffic Engineering instantiated on IPv6 data plane (SRv6)
- A Segment List can be:
- An ordered list of IPv6 Addresses that realises source routing.
    - Segment list is encoded in a routing header called the Segment Routing Header (SRH) – see encoding example in table 2 below.
    - IPv6 Header Destination Address (DA) is the router at the end of a given segment
        - Only router whose address matches packet Destination Address can examine the SRH.
    - SRH contains a segment list comprising of an IPv6 Address of each router in the path.
    - Segments left – denotes the destination address of the next segment
        - IPv6 Header destination address is set to value of segment address
        - *Segments left* value determines which segment in the list is chosen.
            - *Segments left* value is decremented at each segment.
        - Egress node is reached when *segments left=0*

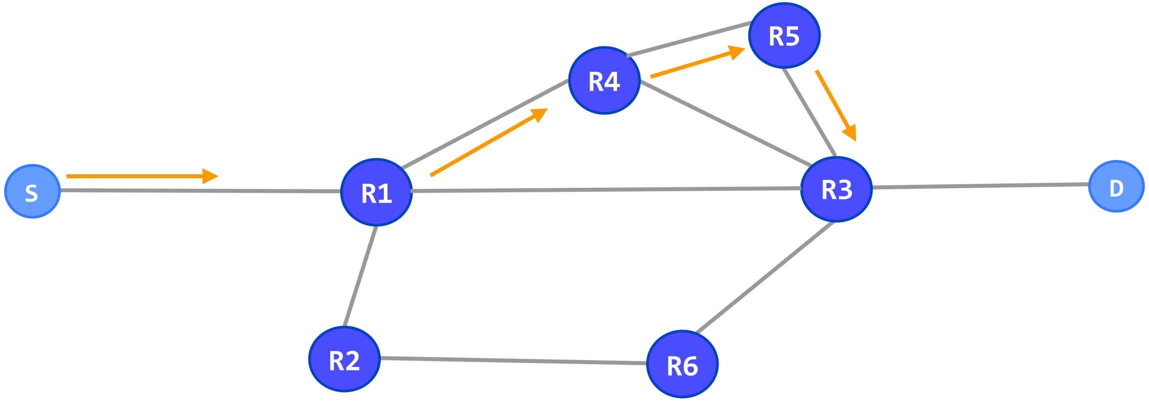
Table 2 – SRv6 Source Routed Path Segment List

Header Type	Parameter encoding.
IPv6 Header	Next header = SRH SA = 2001::101:: DA= 2001:db8:aaaa:111:0:1000::
Segment Routing Header (SRH)	Segments left = 2 Segment 0 – 2001:db8:aaaa:102:0:1000:: Segment 1 – 2001:db8:aaaa:112:0:1000:: Segment 2 – 2001:db8:aaaa:111:0:1000:: Next Header = IP

SRv6 Segment Routing Header (SRH)

Encoding segment endpoints into an SRH

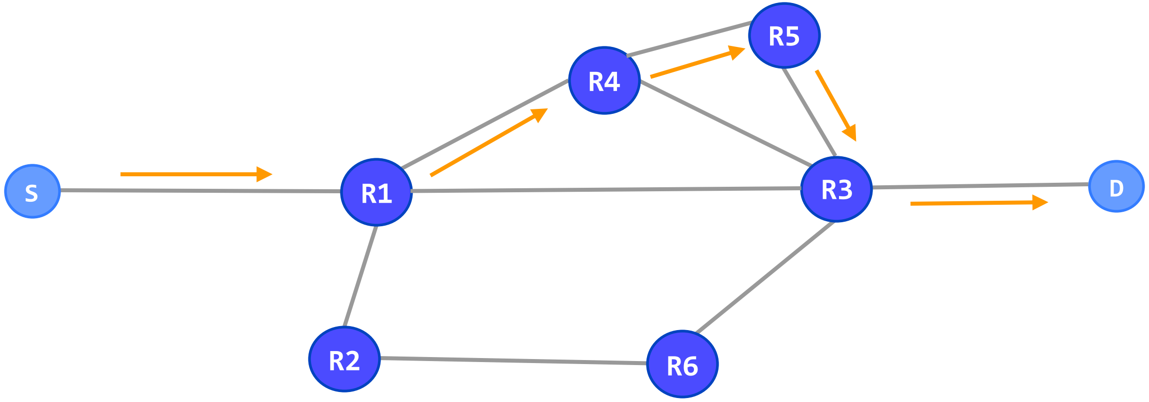
- When there is a need to encode more than a single segment an SRH must be used.
- We can use a segment list in an SR-Policy to create a certain path through the network



SRv6 Segment Routing Header (SRH)

Encoding segment endpoints into an SRH

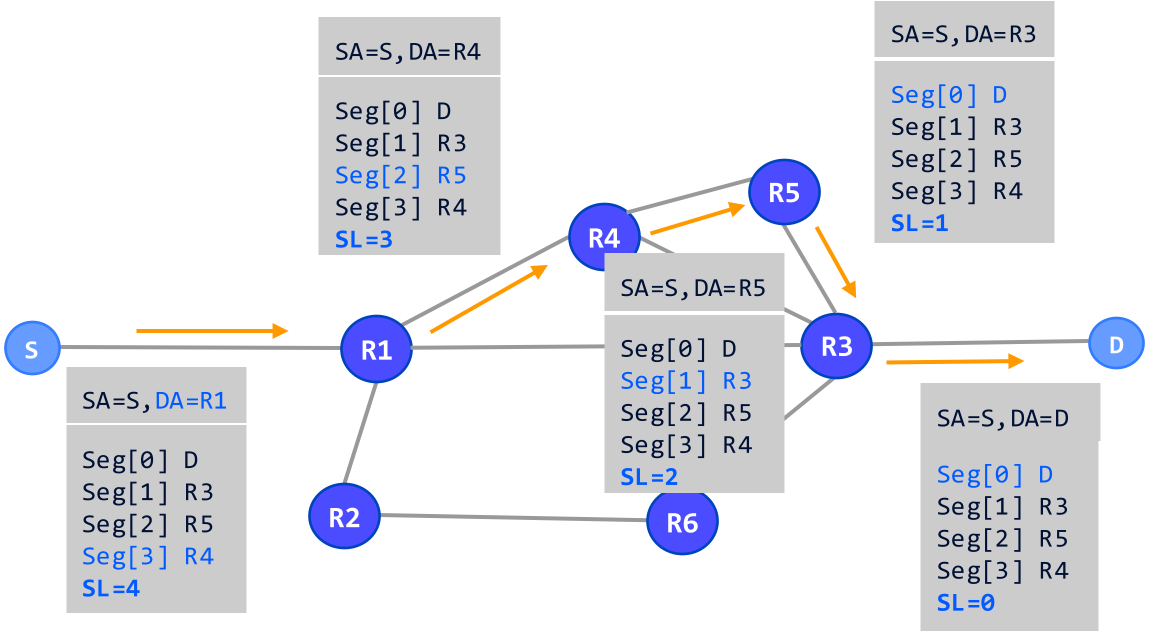
- Let's create a path from S to D using the strict path: R1, R4, R5, R3, D



SRv6 Segment Routing Header (SRH)

Encoding segment endpoints into an SRH

Let's create a path from S to D using the strict path: R1, R4, R5, R3, D



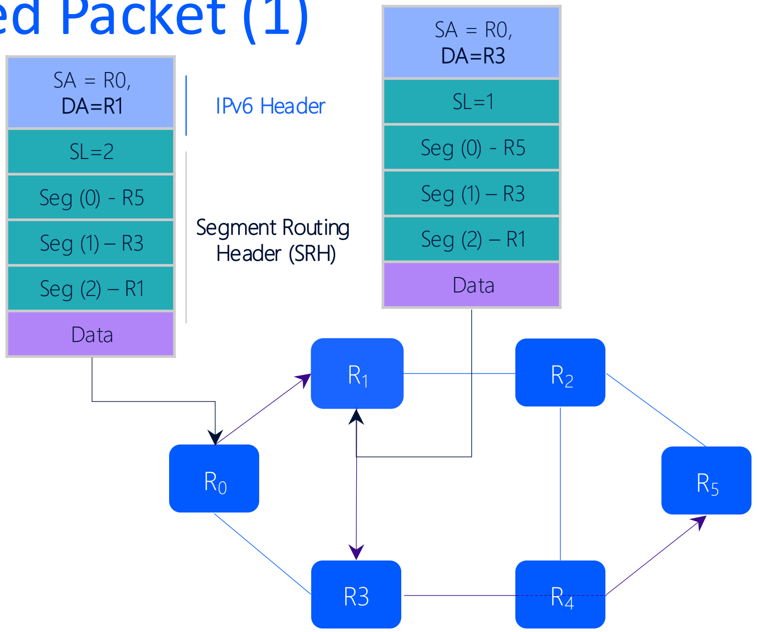
`SRH with 4 SIDs = 4x16 bytes of overhead`

NOKIA

Data Forwarding of SRv6 Encapsulated Packet (1)

Using IPv6 Segment IDs

- SR over IPv6 Data Plane - Path R_0 to R_5 – Via Hops $R_1 \rightarrow R_3 \rightarrow R_5$
- Path R_0 to R_1
  - Illustrative stacks represent packet at router egress
- R_0 will tunnel packet towards R_5
  - SRH constructed with Segment List at R_0
    - Segment List containing IPv6 SIDs associated with each hop e.g. End SID
      - Encoded with destination router as top Segment in list – segment zero.
      - First segment endpoint is last segment in list – segment (2) in this example
      - *segments left (SL)* is set to value matching highest segment number (2) – SL=2
    - SRH is only interrogated by routers with DA = local address
      - IPv6 Source Address (SA) is set to local IPv6 address of R_0.
      - IPv6 Header Destination Address is set to segment list entry indexed in the *segments left* field – in this case, R1.
      - Packet is forwarded towards R_1.



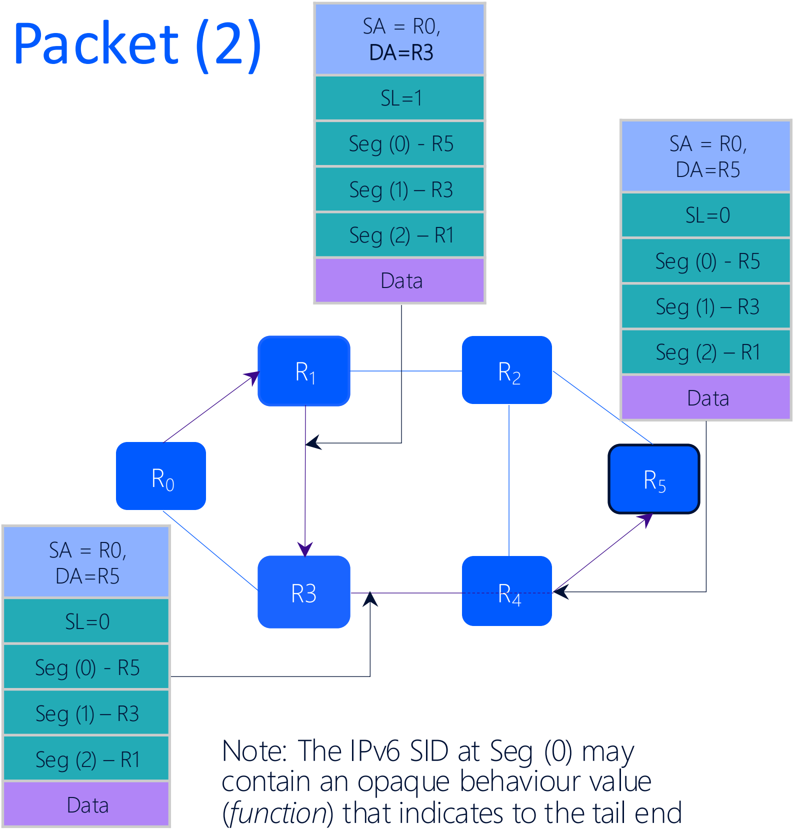
- At R_1
  - Destination address in IPv6 Header is R1.
    - R1 removes IPv6 header and interrogates SRH
    - In SRH Segments left =2. Decrement to SL=1 – Segment(1) = R3.
    - Copy Segment ID value Segment (1) to IPv6 DA
    - Forward packet towards R_3.

  | Public

Data Forwarding of SRv6 Encapsulated Packet (2)

Using IPv6 Segment IDs

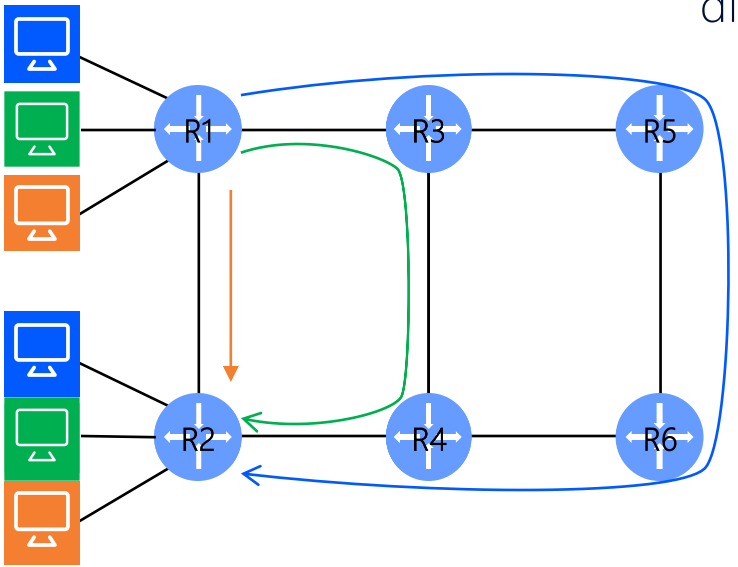
- Path R_0 to R_5 – Via Hops $R_1 \rightarrow R_3 \rightarrow R_5$
- Path R_3 to R_5
  - Packet arrives at R_3
    - Destination address in IPv6 Header is R_3.
      - R_3 removes IPv6 header and interrogates SRH
      - In SRH Segments left =1. Decrement to SL=0 – Segment(0) = R_5.
      - Copy Segment ID value Segment (0) to IPv6 DA. DA=R5
      - Forward packet towards R_4.
  - At R_4
    - Destination address in IPv6 Header is R_5.
      - SRH is **not** interrogated (DA is not R_4) Forward packet towards R_5.
  - At R_5
    - Destination address in IPv6 Header is R_5.
      - R_3 removes IPv6 header and interrogates SRH
      - In SRH Segments left =0.
      - Remove SRH and send for **further processing**



Note: The IPv6 SID at Seg (0) may contain an opaque behaviour value (*function*) that indicates to the tail end router that further processing is required, e.g. VPRN table lookup

Flex Algo

Calculating Optimum Paths Using Different Metrics



Different applications may benefit from following optimum paths calculated using different metrics.

Legend
- Flex Algo 1 (lowest delay)
- Flex Algo 2 (lowest IGP cost)
- Flex Algo 3 (lowest monetary cost)

The Goal of Flexible Algorithms

Using Flexible Algorithms is a simple way to achieve traffic engineering for simple use-case scenarios.

1. For each of multiple applications, optimum paths can be calculated using a different metric, including IGP cost, TE cost or link delay.
2. Separate network topologies can be created and reserved to be used by different applications.
3. A combination of 1 and 2.

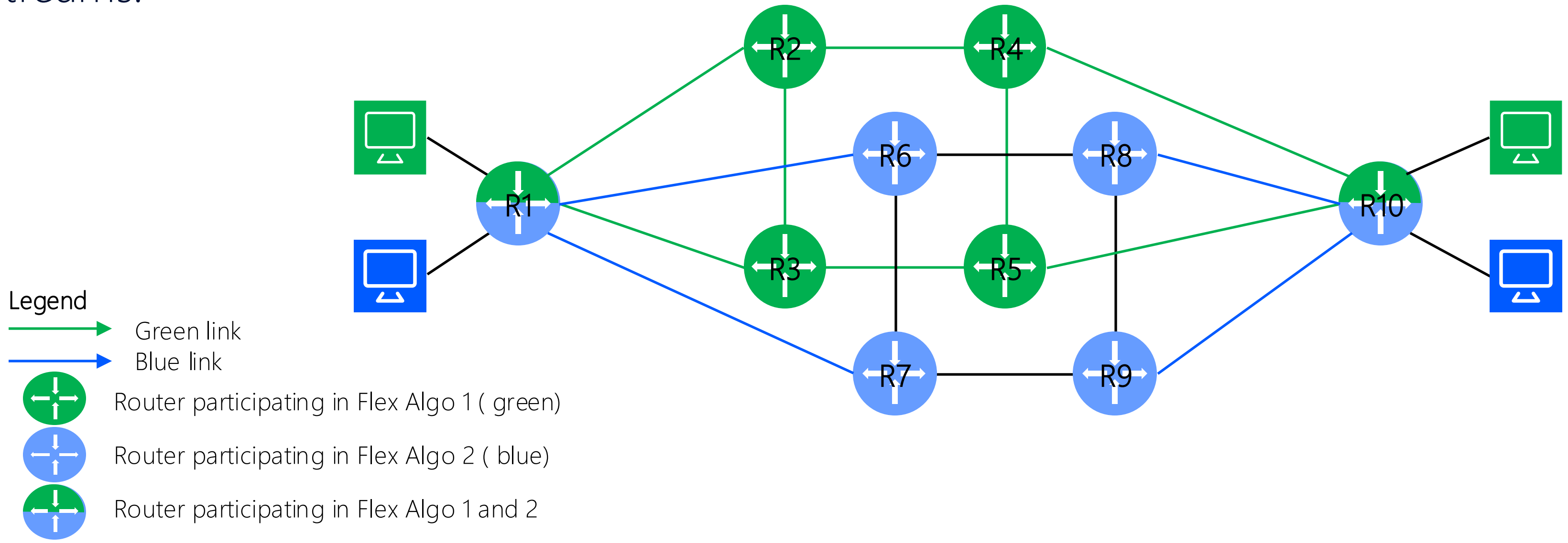
NOKIA

The Goal of Flexible Algorithms [cont.]

- When traffic is steered to be carried by links in a specific network topology, the following is achieved:
  - The traffic is carried over an end-to-end path that has been optimized using a metric that best suits the application.
  - The traffic is carried over links that are reserved for that application, avoiding interference from other competing applications.
- Similar goals are achieved with SRv6 policy, but the advantage of using Flexible Algorithms is that there is no need to configure individual policies.
- SRv6 tunnels will be automatically created by the routing protocol between every pair of routers participating in a given Flexible Algorithm instance.
- An additional advantage of Flexible Algorithms is that a single SID is enough to represent each TE-constrained path, contrary to SRv6 policy which typically requires multiple SIDs.

Creating Separate Network Topologies for Different Applications

Network performance may benefit from reserving some links to be used only by certain applications, to avoid interference from competing traffic streams.



Basics of SRv6

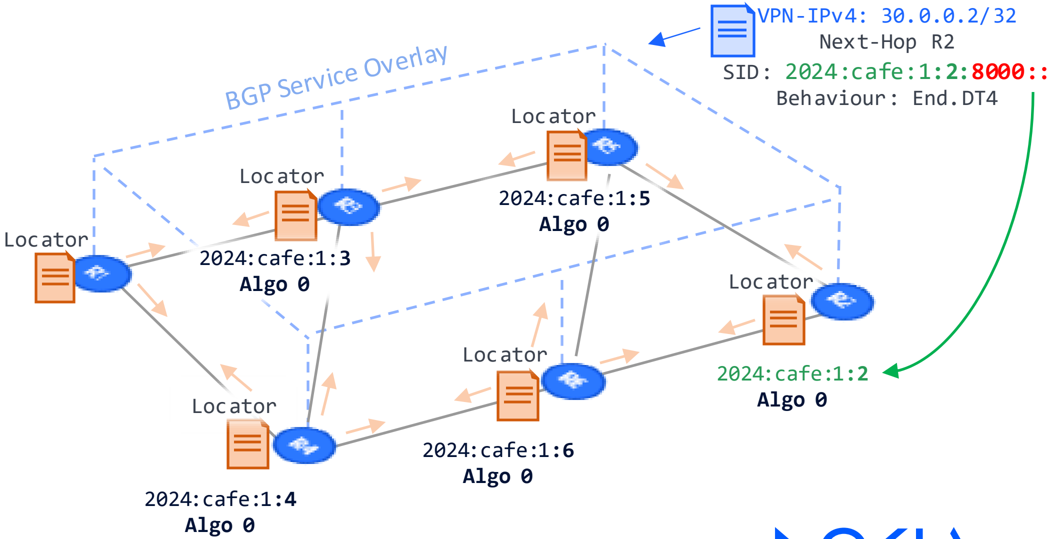
BGP service overlay

SRv6 BGP Service Overlay

Signalling BGP-based services with SRv6

- BGP overlay services are extended for signalling over an SRv6 core
- SRv6 L2 and L3 Service TLVs are defined to encode:
  - The SID value
  - Endpoint behaviour
  - and information about SID structure.

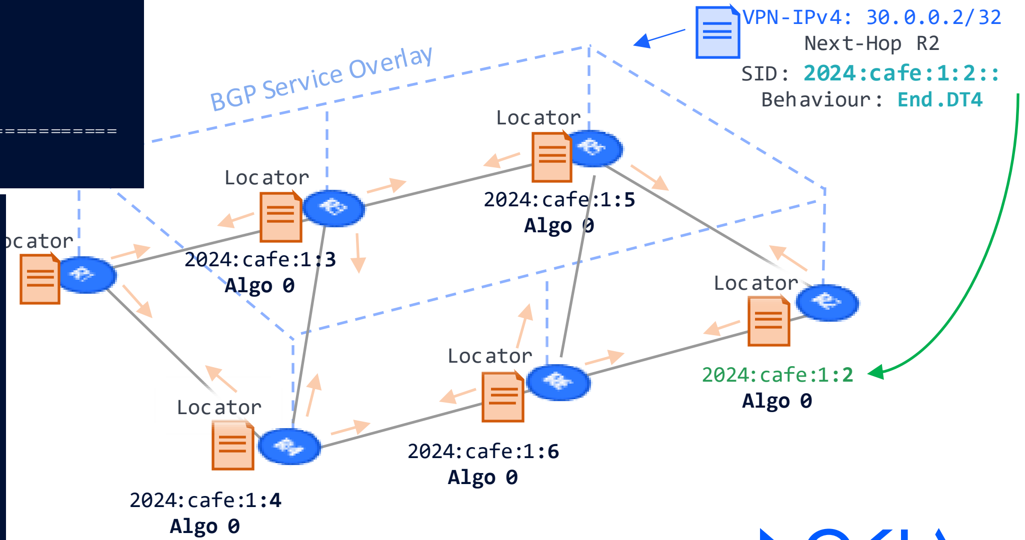
- Can have direct association to a Flex Algo.



SRv6 BGP Service Overlay

```
A:admin@r1# show router 2 route-table

===============================================================================
Route Table (Service: 2)
===============================================================================
Dest Prefix[Flags]                            Type    Proto     Age         Pref
      Next Hop[Interface Name]                                    Metric
-------------------------------------------------------------------------------
30.0.0.1/32                                   Local   Local     01d01h03m   0
       lo0                                                        0
30.0.0.2/32                                   Remote  BGP VPN   00h11m13s   170
       2024:cafe:1:2:8000:: (tunneled:SRV6)                       31
-------------------------------------------------------------------------------
No. of Routes: 2
Flags: n = Number of times nexthop is repeated
       B = BGP backup route available
       L = LFA nexthop available
       S = Sticky ECMP requested
===============================================================================
```

```
A:admin@r1# show router bgp routes 30.0.0.2/32 vpn-ipv4 detail
…

Network          : 30.0.0.2/32
Nexthop          : 192.0.2.2
Route Dist.      : 2:2                     VPN Label       : 524288
…
SRv6 TLV Type    : SRv6 L3 Service TLV (5)
SRv6 SubTLV      : SRv6 SID Information (1)
Sid              : 2024:cafe:1:2::
Full Sid         : 2024:cafe:1:2:8000::
Behavior         : End.DT4 (19)
SRv6 SubSubTLV   : SRv6 SID Structure (1)
Loc-Block-Len    : 48                      Loc-Node-Len    : 16
Func-Len         : 20                      Arg-Len         : 0
Tpose-Len        : 20                      Tpose-offset    : 64
```

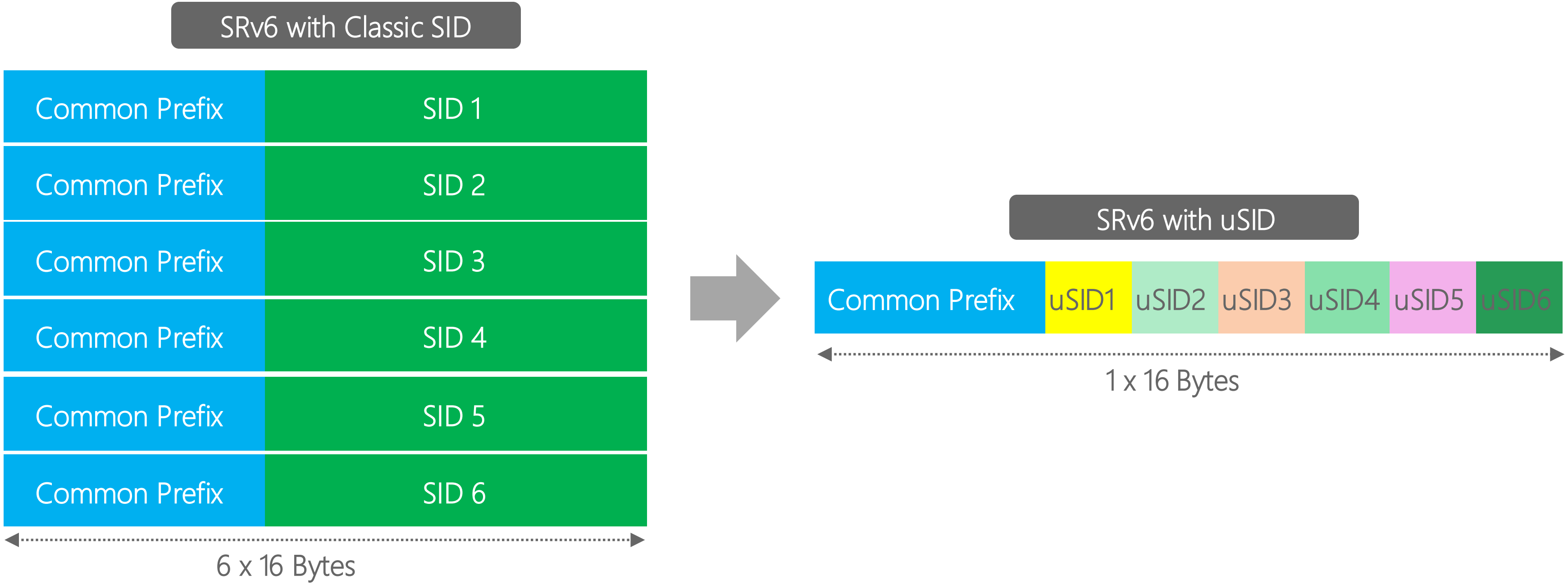


Basics of SRv6

SID compression

SRv6 Header Compression

Micro SID (uSID) Implementation

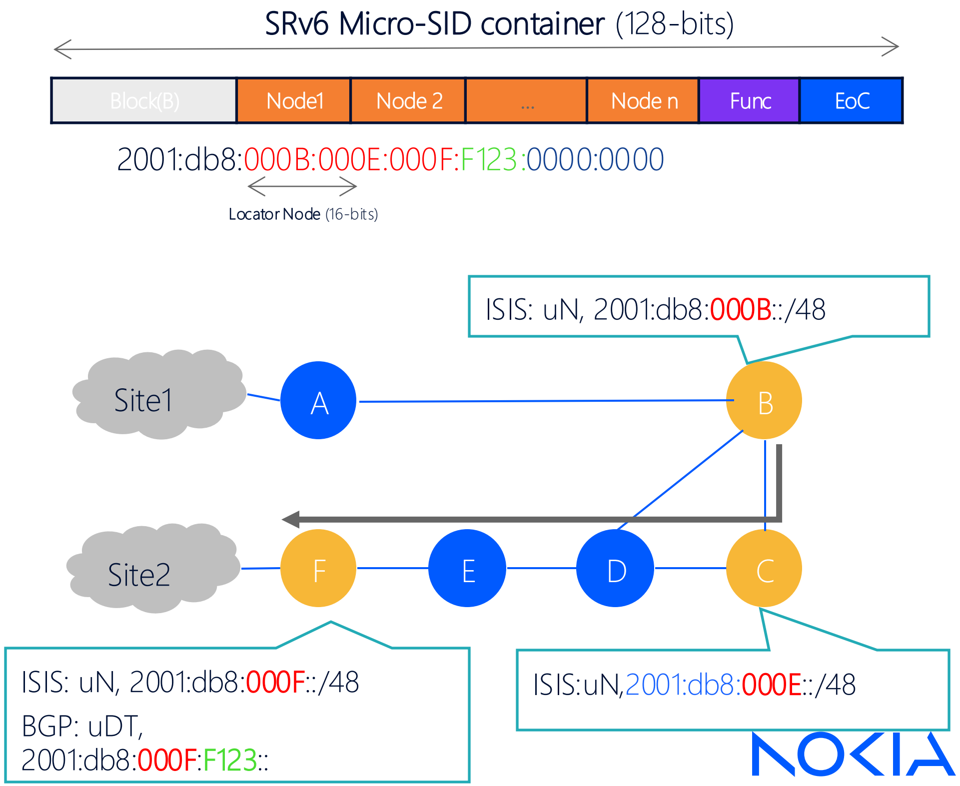


- uSID allows concatenating SRv6 segment identifiers in 128-bit IPv6 Destination Address.
- Reduces packet overhead and increases forwarding efficiency.

SRv6 SID compression and micro-segment identifier (uSID)

SRv6 Micro-SID is commonly supported SRv6 compression

- SRv6 header compression avoid repetition of block field for multi-segment CSPF path
- IPv6 header DA: Micro-SID container
  - uSID block
  - Multiple uSID
  - End of container uSID
- uSID length of 16bits (2Byte) is typical
  - 65,534 non-zero values
  - Split into GIB and LIB (global and local)
- SRH header may have additional Micro-SID containers
- Processing involves multiple lookups and shift/replace operation

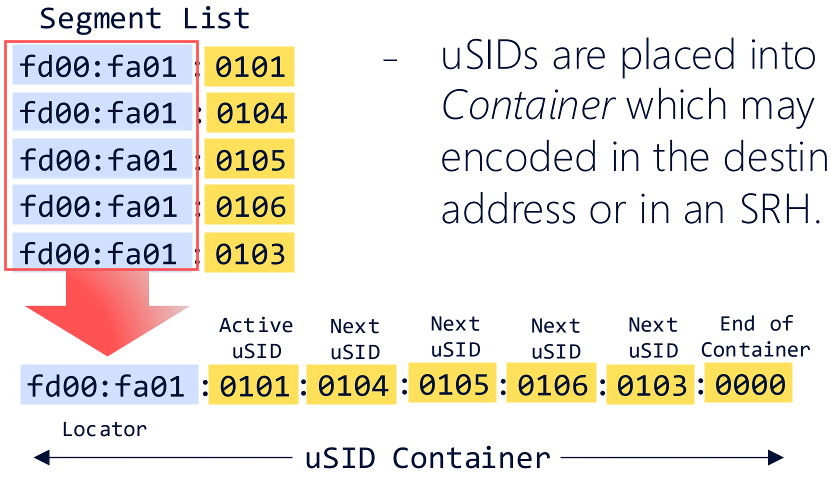


NOKIA

Compressed-SID

uSID (NEXT-CSID) purpose and structure

- Using an SRH containing classic SRv6 SIDs is not ASIC friendly and can impose unreasonable overhead on a packet.
- Micro-SID (uSID) is a flavour of *compressed SID* (CSID) and attempts to overcome both issues.
  - When a sequence of consecutive SIDs in a segment list use the same Locator, compression can be achieved by avoiding repetition of the Locator block.



  - uSIDs are placed into a *uSID Container* which may be encoded in the destination address or in an SRH.

