RIPE NCC
RIPE NETWORK COORDINATION CENTER

SRv6 from idea to operations

Alex Semenyaka | September 2026 | HRNOG

TCP/IP approach

Uniform protocol, distributed brain

- Same packets in any part of the network
- Smart intermediate devices (routers)
- Independent decisions
- Eventual convergence

Challenges

But not all network elements are equal…

- Different types of traffic
- Different application requirements
- Different types of channels
- *Usually* it is not an issue
  - But not *always*

History

Pre-defined routing

Why?

- Necessity of the simple and explicit Traffic Engineering (TE)
  - Better capacity planning
  - Rich routing metrics
  - Eliminating local route fluctuations during recalculations
- Need to better partition and structure the network
  - Robust FRR
- Effortless management and diagnostics with explicit path control

The Dark Ages

Taking swings at the pre-defined routing

- IPv4: Option LSRR and SSRR
  - Source routing
  - Existed from the beginning of IPv4
  - Had security and performance issues

The Dark Ages

Taking swings at the pre-defined routing

- IPv4: Option LSRR and SSRR
  - Source routing
  - Existed from the beginning of IPv4
  - Had security and performance issues
  - *Deprecated and disabled by default at early 2000s*

The Dark Ages

Taking swings at the pre-defined routing

- IPv6: Routing Header type 0 (RH0)
  - Source routing
  - Existed from the beginning of IPv6
  - Had the security issues

The Dark Ages

Taking swings at the pre-defined routing

- IPv6: Routing Header type 0 (RH0)
  - Source routing
  - Existed from the beginning of IPv6
  - Had the security issues
  - *Deprecated in 2007*

The Dark Ages

Taking swings at the pre-defined routing

- Path Computational Elements Approach
  - Predefined routing
  - Invented in 2006
  - Moving the Constrained Shortest Path First (CSPF) onto the external controller with the unified Traffic Engineering Database
    - And then this controller can instruct routers

The Dark Ages

Taking swings at the pre-defined routing

- Path Computational Elements Approach
  - Predefined routing
  - Invented in 2006
  - Moving the Constrained Shortest Path First (CSPF) onto the external controller with the unified Traffic Engineering Database
    - And then this controller can instruct routers
  - *Limited usage (MPLS RSVP-TE and GMPLS)*

The Dark Ages

Taking swings at the pre-defined routing

- OpenFlow SDN
  - Predefined routing
  - Invented in 2008
  - Complete physical separation of the control plane from the data plane
    - From posh and smart branded boxes to cheap stupid white boxes

The Dark Ages

Taking swings at the pre-defined routing

- OpenFlow SDN
  - Predefined routing
  - Invented in 2008
  - Complete physical separation of the control plane from the data plane
    - From posh and smart branded boxes to cheap stupid white boxes
  - *It had been written off by the industry by 2018 due to internal conflicts*
    - The need for a massive TCAM, and NP instead of ASIC
    - Control Plane latency and the responsiveness issue
    - The complexity of versioning multi-level tables

The Renaissance

MPLS-based Segment Routing

- The very first successful and universal [enough] solution
- Invented in 2013
- Moved away from the micromanagement
  - In favor of a combination of distributed and pre-defined decision-making
- Moved beyond the traditional routing paradigm
  - And redefined the meaning of labels in MPLS

The Modern Era

Segment Routing in IPv6

- Adaptation of core concepts from SR/MPLS
- Simplified deployment
- Easy adaptation
  - Not without drawbacks

MPLS

A short recap

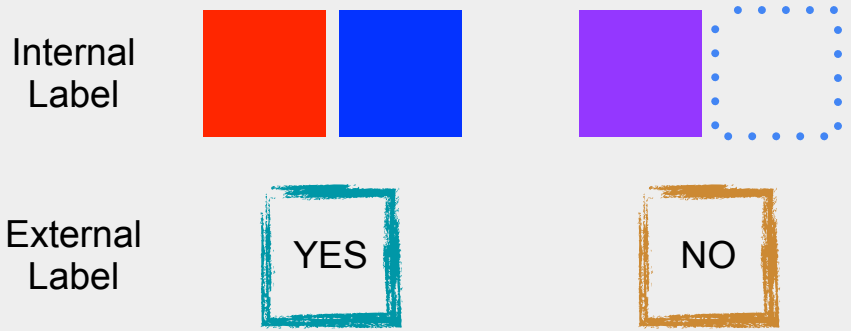
- The **simple** concept of hierarchical labelling of IP packets
- At each level, a single label groups together all “equivalent” packets
  - Forming classes of equivalence
  - The logic behind equivalence is arbitrary and is determined by the network architect
- Labels have the local meaning
- Levels are stacked

A twisted example

- The deepest label: the color of an IP packet
  - Red
  - Blue
  - Purple
  - Transparent (oops)

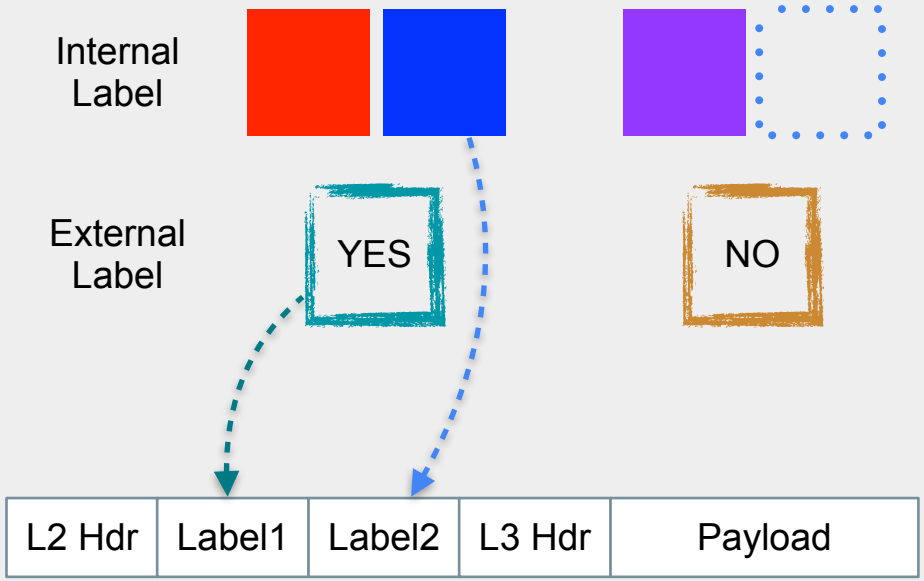
A twisted example: two levels

- The deepest label: the color of an IP packet
  - Red
  - Blue
  - Purple
  - Transparent (oops)
- The topmost level: the necessity
  - At the level of the meaning of life (or "YES")
  - Better not to ever meet ("NO")

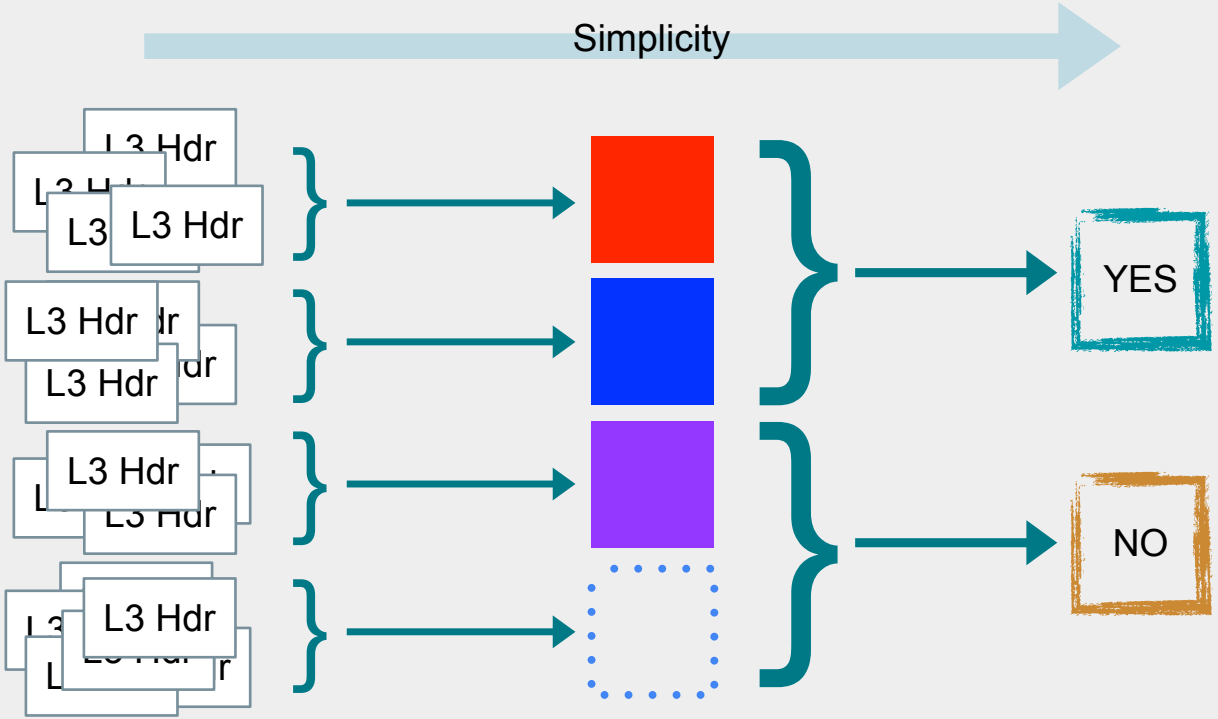


A twisted example: two levels

- The deepest label: the color of an IP packet
  - Red
  - Blue
  - Purple
  - Transparent (oops)
- The topmost level: the necessity
  - At the level of the meaning of life (or "YES")
  - Better not to ever meet ("NO")



A twisted example: two levels



A short recap (cont.)

- The **simple** concept of hierarchical labelling of IP packets
- At each level, a single label groups together all “equivalent” packets
  - Forming classes of equivalence
  - The logic behind equivalence is arbitrary and is determined by the network architect
- Labels have the local meaning
- Levels are stacked
- A router works with the topmost label
- Label switching instead of routing

- And then we pile an **over-engineered control plane** (BGP, LDP, RSVP-TE…) onto this

Segment Routing Approach

The initial concepts

- Labels are no longer local
- Labels are not overwritten
- A label represents a network entity: **a segment**
  - **Label value = Segment ID (SID)**
- These entities (segments) may be of different kinds
- And a label stack is the sequence in which these segments pass through

What on earth are these *entities*?

- The simplest ones are **nodes** and **links**
  - For example, a SID sequence “NodeA, NodeB, LinkX” means “go from Node A to Node B by whichever route they prefer, and then to the next node taking exactly Link X”
  - If we use only loopback addresses, the basic structure becomes very similar to RH0
- But segments can also be more complex
  - **Any operation** that could be performed on an IP packet on the given router can be referred to as a segment
    - For example, directing the packet to the NAT64 box, into a tunnel, or to the scrubbing center

Programmatic network

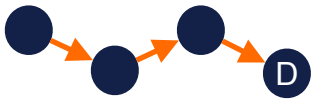
- Each router can work with different segments
  - These router's **functions** should be known in advance
- Then, selecting a **segment** means choosing the instruction (function) which is to be done with the packet at that step
  - Therefore, **a sequence of segments is a program**
- **Function** is what is encoded in SID. **Behaviour** is the local action that the node has associated with this function.

Segment Routing: functions

Some basic examples

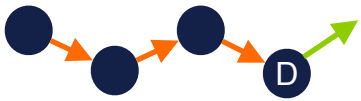
End Instruction

the shortest path to a given node



End.X Instruction

the shortest path to a given node followed by a forwarding over a specific link



End.DT4

the shortest-path to a selected egress node followed by a IPv4-VPN table lookup

End.DT6

the shortest-path to a selected egress node followed by a IPv6-VPN table lookup

More Segment Routing concepts

SR Domain

- Segment Routing Global Block (SRGB): the range of labels reserved for SR-MPLS
  - Other labels are handled outside the SR-MPLS logic
  - Can be combined with the traditional MPLS
- **The part of the infrastructure that supports Segment Routing is called an SR Domain**

Segment Routing Domain

A key operational concept

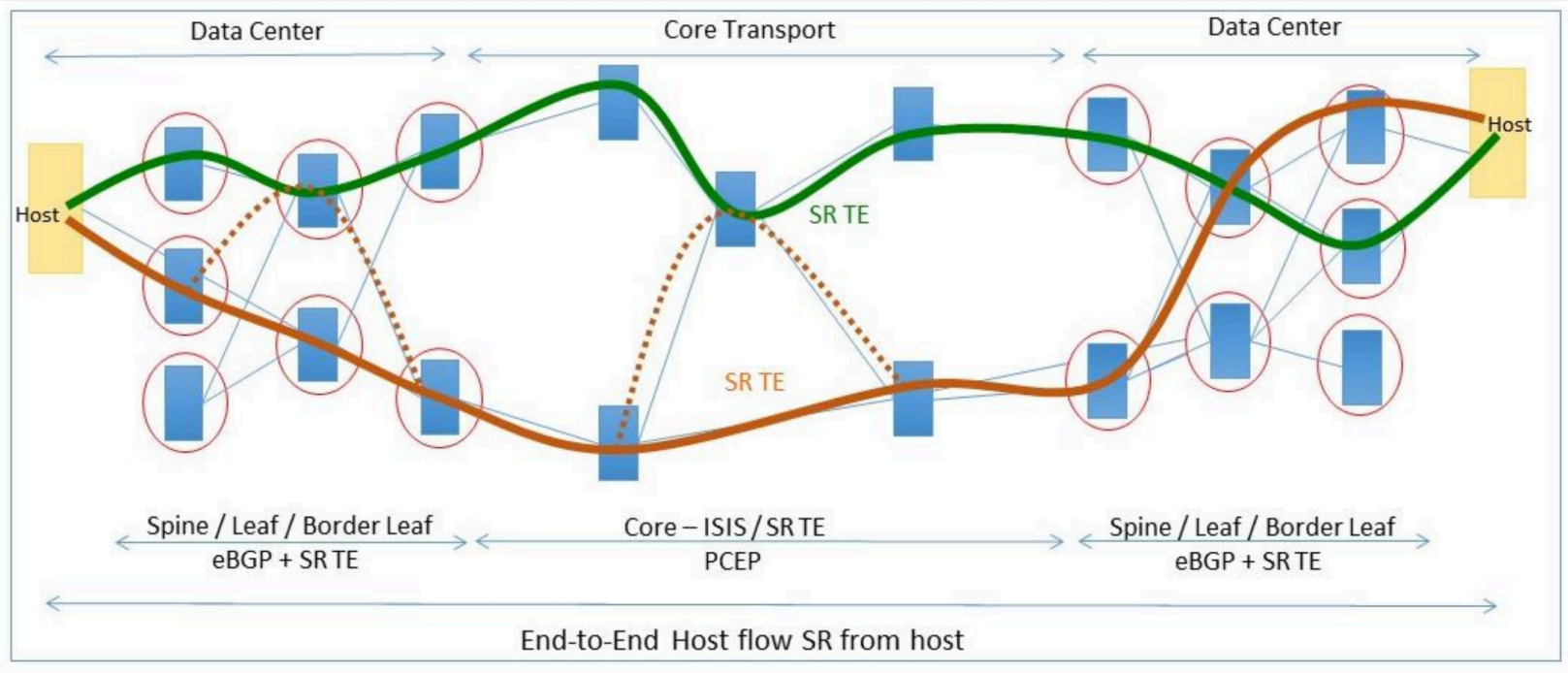
- SR Domain does not introduce any theoretical approaches
- SR Domain is not a new abstraction with its new rules
- It is a **purely operational** concept
  - However, it is a **cornerstone** of any kind of Segment Routing
  - *(we'll discuss it later)*

Segment Routing implementation

The missing part: sources of SIDs

- Statically pre-configured SIDs
- IGP with extensions (“I have a loopback address 10.0.0.1 and it is End SID=10001”)
  - IS-IS
  - OSPFv3
- SDN Controller
  - Collect data: still IGP and/or BGP-LS
  - Instruct routers: PCEP/NETCONF etc
- …or any mix of them.

Example



by Bell Canada (2016), via https://www.segment-routing.net/images/lightreading_report.pdf

SRv6

SR/MPLS Issues

Why the deployment could be challenging

- You need to have an MPLS infrastructure in place everywhere
- All your equipment must support SR/MPLS
- ASICs do not support deep label stacks
  - Scalability would demand some quirks and tweaks

Ready-to-be-used

- IPv6 addresses are large
- RH0 is deprecated, but we still do have a Routing Extension Header
- Which is large too
- And is easy to use
- And we always will have a backup
  - Default routing based on the Basic Header

Translate MPLS into IPv6

- We introduce Segment Routing Header (SRH)
  - It has Type 4 (RH4)
- The SIDs are coded now by 128-bits IDs
  - They are **valid IPv6 address** inside your numeration but with some additional properties

128-bit SRv6 SID

Locator: the route to the node performing the function

Function: the forwarding behaviour to be executed by the node

Arguments (optional): if the function needs (not shown in the drawing)

SRH (RH4) Structure

SRH Base (first 8 octets)			
Bits 0-7	**Bits 8-15**	**Bits 16-23**	**Bits 24-31**
Next Header 8 bits	Hdr Ext Len 8 bits	Routing Type 8 bits	**Segments Left** 8 bits
Last Entry 8 bits	Flags 8 bits	Tag 16 bits	

Segment List area
Segment List[0] (128-bit IPv6 address)
...
Segment List[n] (128-bit IPv6 address)

Optional TLV area (variable length)
TLVs start after Segment List[n]
Present when Hdr Ext Len > (Last Entry + 1) * 2

Routing Type:
- 4 for SRH

Flags:
- O-flag (0x20): OAM support

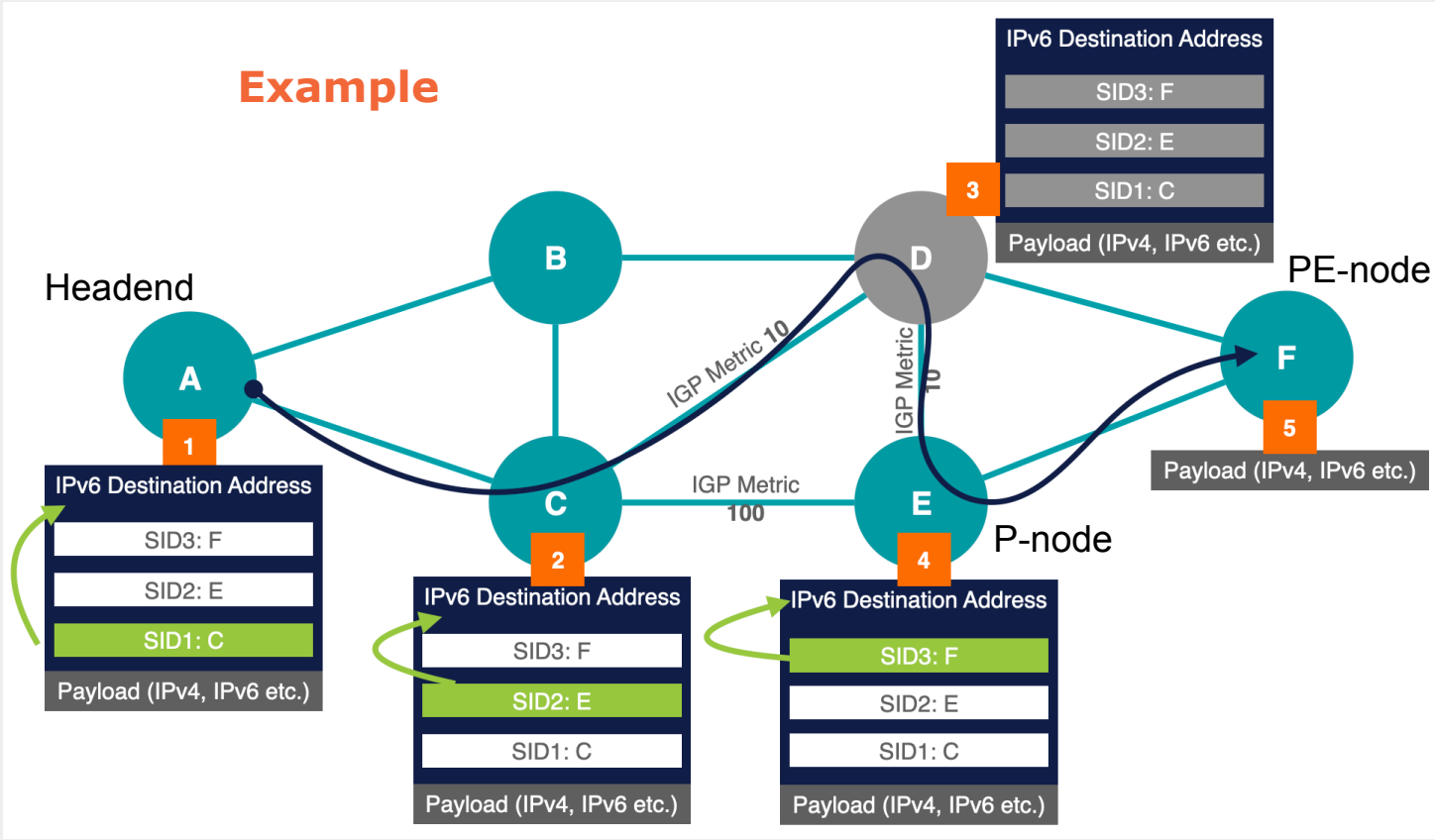
Generic SRH TLV format		
Bits 0-7	**Bits 8-15**	**Bits 16...**
Type 8 bits	Length 8 bits	Variable-length data
Type MSB: 0 = immutable TLV data, 1 = mutable TLV data		

SRv6 vs SR/MPLS

Differences

- We will use IPv6 addresses instead of MPLS labels (obviously)
- Copying the next SID into the Destination IP field of the basic IPv6 header
- Not popping SIDs from SRH, just move the pointer (the Segments Left field)
  - If the segment list is consumed, the SR domain is over
    - But we still can keep SRH in place
    - Or can drop it off if we want
- Network functions can have arguments (optional)
- **Intermediate routers do not need any SRv6 support**
  - Non-SRv6 node forwards the packet as usual, based on the current DA and LPM
- We can either **insert** SRH into the packet, or **encapsulate** entire packet into IPv6/SRH one

SRv6 traversal



Adding SRH into the packet

How do we add SRH?

- **Usually we don't insert**; instead, **we encapsulate** the original packet into the new one, adding both SRH and a basic IPv6 header (T.Encap / H.Encap)
  - Insertion is theoretically possible if the headend is the end host (H.Insert)
    - But in practice it is not recommended ("The encap mode should be used, unless you know what you are doing")
  - Insertion on a transit node (T.Insert) is prohibited
    - Or "acceptable only within SRv6 domain"
- And that's why the support of O-bit in SRH is important
  - A customer's traceroute sees the domain as one hop
    - Unless hop-limit propagate mode and ICMP relay at the headend are enabled
  - The good news is that all major manufacturers usually support O-bit

Overview of SRv6 pre-defined functions (RFC 8986 and its drafts)

End host

- **H.encap**
  Encapsulate created IP-packet in an external IPv6 packet with an SRH
- **H.insert**
  Insert SRH into the created IPv6 packet
- **H.insert.red***
  Insert reduced SRH into the created IPv6 packet

Headend router

- **T.insert**
  Insert SRH into the incoming IPv6 packet **(prohibited)**
- **T.encap**
  Encapsulate incoming IP-packet in an external IPv6 packet with an SRH
- **T.encaps.red***
  Encapsulate incoming IP-packet in an external IPv6 packet with reduced SRH
- **T.l2encap**
  Encapsulate incoming Ethernet-frame in an external IPv6 packet with an SRH
- **T.l2encaps.red***
  Encapsulate incoming Ethernet-frame in an external IPv6 packet with an reduced SRH

Inside SRv6 domain

- **End**
  Send to the next node segment (lookup)
- **End.X**
  Send to the next adj segment (no lookup)
- **End.T**
  Send to the next node segment (lookup in the VRF table specified)
- **End.DX4**
  [SL=0 or no SRH required] Decap and send to the IPv4 adj specified
- **End.DX6**
  [SL=0 or no SRH required] Decap and send to the IPv6 adj specified
- **End.DT4**
  [SL=0 or no SRH required] Decap and send to DA with IPv4 lookup in the VRF table specified
- **End.DT6**
  [SL=0 or no SRH required] Decap and send to DA with IPv6 lookup in the VRF table specified
- **End.B6.Encaps**
  Prepare SRH for the next hop but encap this packet into an extra IPv6 sheath with a special SRv6 delivery policy
- …

* "Red" stands for "reduced": SRH is created without the first segment

Behaviour details

Basic optimisations

- T.encap.red
  - Reduction: we know the next segment, why to add it to the list?
- End – the default behaviour of a SID: "I am a node on the list; proceed to the next segment". However, it may have flavors:
  - Ultimate Segment Decapsulation (**USD**): The last node in the list removes the entire external IPv6 header, including the SRH, and processes the internal packet
  - Penultimate Segment Pop (**PSP**): The penultimate node in the list deletes the SRH after copying the last SID to the DA, but keeps the outer IPv6 header
  - Ultimate Segment Pop (**USP**): the final node removes only the SRH (no decap), and then processes the packet
    - Use-case: end-to-end SRv6 with H.insert at the ends
- Traditional combination: PSP on the penultimate node (P) + USD on the last node (PE)

Nodes taxonomy

Node capabilities may differ

- Headend
  - The point where SRHs are generated, the entry to the SR Domain
  - Must be able to encapsulate
  - Must support external signalling
    - E.g. VXLAN/EVPN
- P router
  - Only an intermediate role
  - May not be able to decapsulate
  - May not support services (End.D*)
- PE router
  - Can decapsulate
  - Understands the necessary services (End.D*)

Make new friends but keep the old

Backbone and datacenter technologies

- P-router, PE-router: well-known concepts from the traditional MPLS approach
  - Similarities: the logic of P and PE
  - Differences: P and PE are no longer topological roles but node capabilities
    - "is this node able to do PSP or decap?" — instead of "is this box a PE?"
- Encapsulation and overlay/underlay networks are familiar from VXLAN/EVPN architecture
  - Similarities:
    - Underlay/overlay model
      - ➡ Underlay: IPv6 routing of locators via an IGP
      - ➡ Overlay: services (e.g., L3VPN/EVPN with corresponding SIDs: End.DT*/DX*/DT2*)
    - Same control plane, same tunnel model, same service semantics — different encapsulation
  - Differences:
    - Underlay and overlay routing are inextricably linked, there is only one namespace rather than two in VXLAN/EVPN
    - The tunnel is neither opaque nor point-to-point — transit nodes are addressable as segments
    - VXLAN is an overlay only; SRv6 can be both overlay and transport

Routing in SRv6

SRv6 routing decisions

Who and how produces SRH?

- Headend – the point where SRHs are generated
- Headend creates an SRH with the sequence of SID based on SR Policy
  - **Explicit** — an exact segment list is specified
  - **Dynamic** — the segment list is calculated based on constraints
  - **Composite** — several policies are combined
- According to RFC 9256, an SR-policy is identified by the triplet (**headend, color, endpoint**)
  - Color is a number that denotes **intent** (‘low latency’, ‘high throughput’, ‘bypass region X’, ‘low-cost transit’)
- SR Policy can produce multiple path candidates
  - SID list corresponds to the best one
    - (define “best”, of course - that’s where the **color** works)
- A trivial policy uses usual Shortest Path Tree (SPT)

Explicit SR Policy

Linux examples

```
# Paths through SID1 (fc00:1::1) and then SID2 (fc00:2::1)

# H.encap
ip sr tunsrc set fd00:cafe:ff::1
ip -6 route add 2001:db8:1::/64 encap seg6 mode encap segs fc00:1::1,fc00:2::1 dev eth0

# H.insert
ip -6 route add 2001:db8:1::/64 encap seg6 mode inline segs fc00:1::1,fc00:2::1 dev eth0
```

Explicit SR Policy

Cisco (IOS XR) example

Hmmm… Now it does not look that simple

```
hw-module profile segment-routing srv6 mode base
segment-routing
 srv6
  encapsulation
   source-address fd00:cafe:ff::1
  locators
   locator MAIN
    prefix fd00:cafe:ff::/48
 traffic-eng
  segment-lists
   segment-list SL1
    srv6
     index 10 sid fd00:cafe:1::1
     index 20 sid fd00:cafe:2::1
     index 30 sid fd00:cafe:3::e006
  policy P1
   srv6
    locator MAIN
   color 100 end-point ipv6 fd00:cafe:3::1
   candidate-paths
    preference 100
     explicit segment-list SL1
router static
 address-family ipv6 unicast
  3fff:77::/48 sr-policy P1
```

Static SR Policy

Juniper, Nokia, Huawei examples

- It takes some ingenuity to set up this configuration
- However, there is no guarantee that it will work correctly: the documentation does not clearly describe this use case

Operational reality: static-route steering into an explicit policy is for labs

Dynamic SR Policy

Simple architecture: no SRv6 controller

- IGP distributes locators and SRv6 topological information
  - The network knows how to find the places where *topological* SIDs live
    - End, End.X
  - The IGP is primarily responsible for underlay reachability and topology
- BGP typically carries *service* SIDs and may carry policies
  - Service SIDs live in the BGP world, particularly when it comes to VPNs and such
    - End.D*
- The headend calculates candidate paths

New functions, old protocols

How to code functions

- Functions are enumerated as Endpoint Behaviour codepoints
  - End = 1, End.X = 5, End.T = 9, End.DX6 = 16, End.B6.Encaps.Red = 27…
  - Registry: IANA, Segment Routing Group
- What if I want some special function?
  - Secretly? Go for 65535 (Opaque)
  - Privately? Feel free to use the range 32768–34815
  - Making everybody happier? Apply for your number in the range
    - First Come First Served, no technical assessment is carried out
- Where these codepoints are used?
  - **Only on the control plane**
  - **They cannot be calculated from SIDs on the data plane**

IGP Support: OSPFv3

OSPFv3 extensions

- Added a **new SRv6 Locator LSA (Type 0xA02A, Function Code 42)** that carries the SRv6 Locator TLV (type 1)
  - The 0/1 algorithm locator should be duplicated in the regular (E-)Intra-/Inter-Area-Prefix and AS-External LSAs for legacy nodes
- Added **new TLVs 8 (SR-Algorithm), 12 (Node MSD) and 20 (SRv6 Capabilities)** to the OSPFv3 Router Information LSA (Function Code 12, Type 0xA00C)
- Added **new sub-TLVs 9 (Link MSD), 31 (End.X) and 32 (LAN End.X)** to the Router-Link TLV of the OSPFv3 Extended-Router-LSA (Type 0xA021, Function Code 33)
- The main structure was created for SR-MPLS and then tailored for SRv6 — except the locator, which got a dedicated LSA instead of a sub-TLV of the prefix LSAs

OSPF extensions for SRv6

New SRv6 Locator LSA (function 42)

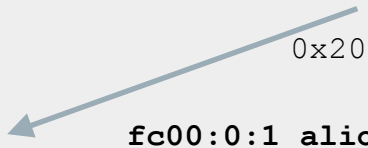
A standard LSA structure

LSA header	20	
LS Age	2	…
LS Type	2	0xA02A (U=1, area scope)
Link State ID (arbitrary unique)	4	…
Advertising Router	4	10.0.0.1
LS Sequence Number	4	…
LS Checksum	2	…
Length	2	72 (LSA hdr + TLV)
TLV 1: SRv6 Locator (may be multiple)		

OSPF extensions for SRv6: SRv6 Locator LSA

TLV 1 (SRv6 Locator)

Type	2	1
Length	2	48
Route Type	1	1 (intra-area)
Algorithm	1	0
Locator Length	**1**	**48**
Flags (PrefixOptions: AC E N DN P - LA NU)	1	0x20 (N=1)
Metric	4	1
Locator	**8**	**fc00:0:1 aligned to 32 bits**
sub-TLV 1: SRv6 End SID	32	



Sub-TLV 1 (SRv6 End SID)

Type	2	1
Length	2	28
Flags (reserved)	1	0
Reserved	1	0
Endpoint Behavior (code)	**2**	**48 (uN)**
SID	**16**	**fc00:0:1::**
Sub-TLV 10: SRv6 SID Structure	**8**	

Sub-TLV 10 (SRv6 SID Structure)

Type	2	10
Length	2	4
Locator Block Length	**1**	**32**
Locator Node Length	**1**	**16**
Function Length	**1**	**0 (for uN)**
Argument Length	**1**	**80 (CSIDs)**

OSPF extensions for SRv6: Router-Link TLV of the OSPFv3 Extended-Router LSA

Sub-TLV 31 (End.X)

Type	2	31
Length	2	32
Endpoint Behavior Code	**2**	**56 (uA)**
Flags (B S P …)	1	0xA0 (B=1, P=1)
Reserved1	1	0
Algorithm	1	0
Weight	1	0
Reserved2	2	0
SID	**16**	**fc00:0:1:e003::**
sub-TLV 30: SRv6 SID Structure	8	

Sub-TLV 32 (LAN End.X)

Type	2	32
Length	2	36
[same Sub-TLV 31 fields]	8	
Neighbor Router-ID	**4**	**10.0.0.3**
SID	**16**	**fc00:0:1:e003::**
sub-TLV 30: SRv6 SID Structure	8	

OSPF extensions for SRv6: Router-Link TLV of the OSPFv3 Extended-Router-LSA

Sub-TLV 9 (Link MSD) to Router-Link TLV

Type	2	9
Length	2	8
MSD-Type / MSD-Value	1+1	41 / 11 (SRH Max SL)
MSD-Type / MSD-Value	1+1	42 / 2 (SRH Max End Pop)
MSD-Type / MSD-Value	1+1	44 / 3 (SRH Max H.encaps)
MSD-Type / MSD-Value	1+1	45 / 8 (SRH Max End D)

sub-TLV 30 (SRv6 SID Structure) of Sub-TLV 31 and 32

Type	2	30
Length	2	4
Locator Block (LB)	**1**	**32**
Locator Node (LN)	**1**	**16**
Function (F)	**1**	**16**
Argument (A)	**1**	**64**

OSPF extensions for SRv6: Router Information LSA (LSA 12)

TLV 8 (SR-Algorithm)

Type	2	8
Length	2	2
Algorithm	1	0
Algorithm	1	128
<padding to 4 bytes boundary>	2	0

TLV 20 (SRv6 Capabilities)

Type	2	20
Length	2	4
Flags (- O …)	**2**	**0x4000 (O=1)**
Reserved	2	0
Sub-TLVs (not defined yet)	0	0

OAM Capabilities

TLV 12 (Node MSD)

Type	2	12
Length	2	8
MSD-Type / MSD-Value	1+1	41 / 11 (SRH Max SL)
MSD-Type / MSD-Value	1+1	42 / 2 (SRH Max End Pop)
MSD-Type / MSD-Value	1+1	44 / 3 (SRH Max H.encaps)
MSD-Type / MSD-Value	1+1	45 / 8 (SRH Max End D)

IGP Support

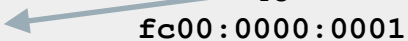
IS-IS extensions for SRv6

- Added a new **TLV 27 (SRv6 Locator)**
  - The 0/1 algorithm locator should be duplicated in TLV 236/237 (IPv6 Reachability) for legacy nodes
- Added a new **sub-TLV 19 (SR-Algorithm), 23 (Node MSD), and Sub-TLV 25 (SRv6 Capabilities)** to TLV 242 (Router Capability)
- Added new **sub-TLVs 15 (Link MSD), 43 (End.X) and 44 (LAN End.X)** to the topological TLV 22, 23, 25, 141, 222 and 223
- The main structure was created for SR-MPLS, and then tailored for SRv6

IS-IS extensions for SRv6

New TLV 27 (SRv6 Locator)

Type	1	27
Length	1	47
IS-IS MTID	2	0 (standard topology)
Locator entry *(may be multiple)*		
Metric	4	1
Flags *(only usual IS-IS flag D is defined)*	1	0x00 (D=0)
Algorithm	1	0 (SPF)
Loc-Size	**1**	**48**
Locator	**6**	**fc00:0000:0001**
Sub-TLV-len	1	31
Sub-TLV 4: Prefix Attribute Flags		
Sub-TLV 5: SRv6 End SID		
Sub-sub-TLV 1: SRv6 SID Structure		
Sub-TLV 1: 32-bit Administrative Tag (not counted in Length)		



IS-IS extensions for SRv6: TLV 27 (SRv6 Locator)

Sub-TLV 4 (Prefix Attribute Flags)

Type	1	4
Length	1	1
Flags (X R N E A ...)	1	0x20 (Node flag **N=1**)

Sub-TLV 5 (End)

Type	1	5
Length	1	26
Flags (reserved)	1	0
Endpoint Behavior Code	**2**	**48 (uN)**
SID	**16**	**fc00:0:1::**
Sub-sub-TLV-len	**1**	**6**

Sub-sub-TLV 1 (SID Structure) of Sub-TLV 5

Type	1	1
Length	1	4
Locator Block Length	**1**	**32 (fc00:0000)**
Locator Node Length	**1**	**16 (:0001:)**
Function Length	**1**	**0 (uN has none)**
Argument Length	**1**	**80 (next uSIDs)**

IS-IS extensions for SRv6: TLV 242 (Router Capability)

Sub-TLV 19 (SR-Algorithm)

Type	1	19
Length	1	2
Algorithm	1	0 (SPF)
Algorithm	1	128 (Flex-Algo)

Sub-TLV 23 (Node depth limits)

Type	1	23
Length	1	8
MSD-Type / MSD-Value	1+1	41 / 11 (SRH Max SL)
MSD-Type / MSD-Value	1+1	42 / 2 (SRH Max End Pop)
MSD-Type / MSD-Value	1+1	44 / 3 (SRH Max H.encaps)
MSD-Type / MSD-Value	1+1	**45 / 0** (SRH Max End D)

(reality of hardware!)

Sub-TLV 25 (SRv6 Capabilities)

Type	1	25
Length	1	2
Flags (- O …)	**2**	**0x4000 (O=1)** ⇦ OAM Capabilities

IS-IS extensions for SRv6: TLV 22 (Extended IS Reachability)

Sub-TLV 15 (Link depth limits)

Type	1	15
Length	1	8
MSD-Type / MSD-Value	1+1	41 / 11 (SRH Max SL)
MSD-Type / MSD-Value	1+1	42 / 2 (SRH Max End Pop)
MSD-Type / MSD-Value	1+1	44 / 3 (SRH Max H.encaps)
MSD-Type / MSD-Value	1+1	**45 / 0** (SRH Max End D) ⇦ (reality of hardware!)

Sub-TLV 43 (End.X SID)

Type	1	43
Length	1	28
Flags (B S P ...)	1	0xA0 (B=1, S=0, P=1)
Algorithm	1	0
ECMP Weight	1	0
Endpoint Behavior Code	**2**	**56 (uA)**
SID	**16**	**fc00:0:1:e001::**
Sub-sub-TLV-len	1	6
Sub-sub-TLV 1	6	<Same structure as with TLV 27>

Dynamic SR Policy with IGP

Cisco example

```
! once, platform-dependent, reload required
hw-module profile segment-routing srv6 mode base

! common SR block
segment-routing
 srv6
  encapsulation
   source-address fd00:cafe:ffff::a
  locators
   locator MAIN
    prefix fd00:cafe:a::/48

router ospfv3 1
 segment-routing srv6
  locator MAIN
 <usual OSPF config>
```

Dynamic SR Policy with IGP

Nokia example

```
# common SR block
configure router "Base" segment-routing segment-routing-v6 source-address fd00:cafe:ffff::a
configure router "Base" segment-routing segment-routing-v6 locator "MAIN" block-length 32
configure router "Base" segment-routing segment-routing-v6 locator "MAIN" prefix ip-prefix fd00:cafe:a::/48
configure router "Base" segment-routing segment-routing-v6 locator "MAIN" admin-state enable

# IGP (IS-IS only — SR OS has no OSPFv3 SRv6)
configure router "Base" isis 0 segment-routing-v6 admin-state enable
configure router "Base" isis 0 segment-routing-v6 locator "MAIN" level 2 admin-state enable
<usual IS-IS config>
```

Dynamic SR Policy with IGP

Juniper example (SRv6 does not support OSPFv3)

```
# common SR block
set routing-options source-packet-routing srv6 locator MAIN fd00:cafe:a::/48

# IGP (IS-IS only — Junos has no OSPFv3 SRv6)
set protocols isis source-packet-routing srv6 locator MAIN end-sid fd00:cafe:a:1:: flavor psp usd
set protocols isis interface <ifd>.0 level 2 srv6-adjacency-segment unprotected locator MAIN
<usual IS-IS config>
```

Huawei

```
# common SR block
segment-routing ipv6
 encapsulation source-address fd00:cafe:ffff::a
 locator MAIN ipv6-prefix fd00:cafe:a:: 48 static 32

# IGP
ospfv3 1
 segment-routing ipv6 locator MAIN
 <usual OSPFv3 config>
```

BGP Support for End.D*

BGP

- It is plain Multiprotocol BGP (MP-BGP, RFC 4760) — the same address families as in MPLS L3VPN/EVPN, plus one new attribute
  - Not BGP-LS (AFI 16388 / SAFI 71), which merely exports link-state topology to a controller
    - nor its EPE extension (BGP Egress Peer Engineering), which describes external peering links beyond the IGP domain
  - Not BGP SR Policy (SAFI 73), which delivers ready-made policies from a controller to headends
- Described in **RFC 9252** (BGP Overlay Services Based on SRv6)

BGP Support for End.D*

SRv6 attribute in BGP

Prefix-SID attribute		
TLV type	1	5 (L3), 6 (L2)
Reserved		
sub-TLV type: SRv6 SID Information	1	
Reserved	1	
SRv6 SID Value	**16**	**fc00:0:1:e004::**
SID Flags (reserved)	1	0
Endpoint Behavior (code)	**2**	**19 = End.DT4**
Reserved	1	0
sub-sub-TLV type 1: SRv6 SID Structure	**6**	
Locator Block Length	1	32 (fc00:0000)
Locator Node Length	1	16 (:0001:)
Function Length	1	16 (:e004:)
Argument Length	1	0 (no arg)
Transposition Length*	1	0 or 16
Transposition Offset*	1	0 or 48

* Used to transfer the part of SID to NLRI; then attribute is the same for different SIDs which allows to pack many prefixes in one UPDATE

BGP Support for End.D*

SRv6 NLRI

Length	1	88 (in bits, label:24 + RD:64 + prefix:24)
Label*	3	[MPLS: 20 bits of value + EXP + BoS]
RD	8	65000:100
Prefix	var	10.1.0.0

* Here the part of SID can be transferred as a "label"

SRv6 MP_REACH_NLRI Attribute

MP_REACH_NLRI	
AFI	1
SAFI	128
Next Hop	2001:db8::1
NLRI	

SRv6 EXTENDED_COMMUNITIES

Route Target	65000:100
Color (RFC 9012, optional)	666

IGP + BGP

Why do we need both?

- We don't, in general. But usually we do.
- **End and End.X family describes the topology**: a node and a link
  - This is the domain of a link-state IGP
  - It knows that the link exists and what its metric is
- **End.D* describes a service** (VRF, CE, etc)
  - IGPs are not aware of them (and don't need to be)
- And **we usually use some IGP** anyway
  - In theory, there is BGP-LS, but in SRv6 it is only used with the controller
    - And anyway, it's a different BGP flavour
- There may also be BGP-only fabrics (RFC 7938-style data centres)
  - Locators are advertised as standard IPv6 prefixes in BGP unicast
  - End SIDs are addresses derived from them
  - There is no separate End.X signalling in this case
  - But in this case, BGP is IGP 🙂

Dynamic SR Policy

SRv6 controller

- In the SRv6 world the controller does not enable SRv6, but makes orchestration and path computation [more] scalable and flexible
- So, the controller needs to be able to see the network topology and SRv6 attributes
  - Typically, via BGP-LS and/or IGP
- The controller then calculates candidate paths that meet the required constraints/demands, and forwards them to the headends
  - For example, via PCEP
    - (Now it is exactly where it needs to be)
  - Or NETCONF
  - Or BGP SR-TE
- The controller does not replace underlay routing, but operates **above** it
- The controller is responsible for determining **which policy** to select and **for which traffic**, and sometimes for **recalculating** it when the network changes

Fast Re-Route

TI-LFA

- SRv6-capable node can detect the outages of an adjacent link or node
  - Standard mechanisms (BFD etc)
- Also it can calculate the alternative paths to destinations
- Such a SRv6-node is called Point of Local Repair (**PLR**)
- If a failure occurs, the PLR immediately begins to encapsulate transit traffic into a pre-computed Segment List (repair path)
  - A **repair list** is a list of SIDs that the PLR *attaches* to a packet in order to reroute it around the failure point
  - It is usually short: 1–3 segments, after which the packet reaches a point from which it can reach the original destination without routing loops
- This way of doing fast re-routing is called Topology Independent Loop-Free Alternate (**TI-LFA**)
- It is usually implemented without a controller, although it is possible to use one

Fast Re-Route: against standards

T.insert in TI-LFA

- The traditional approach to attach a repair list is to encapsulate it one more time into IPv6+SRH, where SRH is formed from the repair list
- It posed some challenges:
  - The decapsulation usually should be performed by a P-router, which may be not able to do it
  - The size of the packet grew too much, challenging MTU capabilities
- But inside the SR domain we already have our own SRH under our control
  - Thus, the insertion (T.insert) of new SIDs for TI-LFA was a popular approach
  - And probably it might be a reasonable exclusion from the rules

SID Compression

CSID and uSID

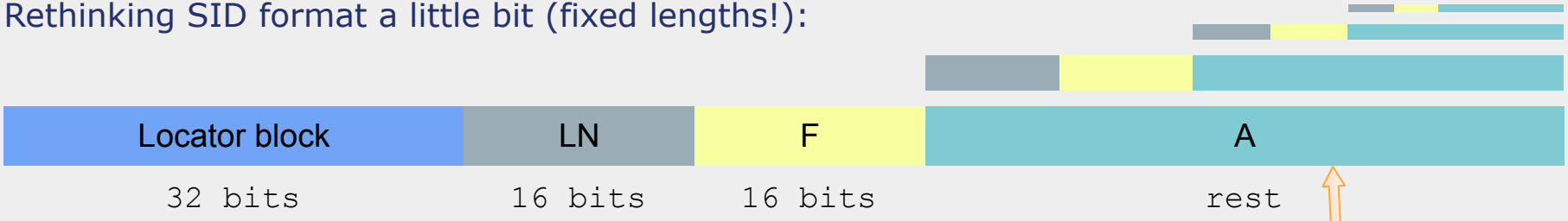
Next optimisation: shrink them

- The goal is to decrease the overhead of SRv6
  - IPv6 basic header + SRH are quite big
- Standardised recently, RFC 9800 (looooong way from the draft in 2019)
- uSID is Cisco's marketing and historical name for Compressed SID (CSID) in the NEXT-CSID approach
- **NEXT-CSID**: Cisco (original uSID)
  - Use DA as a storage for SIDs (CSIDs are packed into one IPv6 address)
  - Might be no SRH at all
- **REPLACE-CSID**: Huawei (Generalized SRv6)
  - SRH is a packed container where CSIDs are stored
  - Only current CSID is in DA

uSID (NEXT-CSID)

128 bits should be enough for everyone

Rethinking SID format a little bit (fixed lengths!):



- Locator-Block (LB): a common prefix for the entire domain (the same for all SIDs)
- Locator-Node (LN): 16-bit SR-node identifier, a **global CSID**
  - 32-bit block + 16-bit CSID = F3216 format (Cisco)
  - For operators without spare /32s, the F4816 also exists
- Function (F): 16-bit **local CSID**
- Argument (A) — all remaining bits
  - Key innovation: the argument of the global CSID consists of the following path CSIDs

uSID container

SR Traversal with uSIDs

Locator block	gCSID1	gCSID2	locCSID2	gCSID3	gCSID4	locCSID4
32 bits	16 bits	16 bits	16 bits	16 bits	16 bits	16 bits

Locator block	gCSID2	locCSID2	gCSID3	gCSID4	locCSID4	0000
32 bits	16 bits	16 bits	16 bits	16 bits	16 bits	

Locator block	gCSID3	gCSID4	locCSID4	0000
32 bits	16 bits	16 bits	16 bits	

Locator block	gCSID4	locCSID4	0000
32 bits	16 bits	16 bits	

end-of-container

uSID lookup

Just our beloved LPM

- `<LB><gCSID>::/48` identifies the node, `<LB><gCSID><locCSID>::/64` identifies the special treatment
  - `gCSID` is **LN**, a node locator
  - `locCSID` is **F**, a function
- The node that carries them has 2+ records in **FIB**:
  - `<LB><LN>::/48` -> uN (same as End in the usual SR)
  - `<LB><LN><F1>::/64` -> special treatment corresponding to the function F1
  - `<LB><LN><F2>::/64` -> special treatment corresponding to the function F2
    - **ASIC** will forward packets for those special treatments
- Up to 6 uSIDs in one DA without needing an SRH

uSIDs address plan

- IETF: 5f00::/16
- Cisco proposes to use ULA-like prefix fcbb:bb00::/32
- Even when you have enough /32, consider ULA
  - ULA is not routable in the global Internet
- GUA can be used, too
  - Not recommended though

No SRH, it is good! But something to remember

- No Segment Routing OAM
- No [theoretical] protection
- Up to 6 uSIDs in one DA
- It is neither T.insert nor T.encap, we are just rewriting DA
  - Important for TI-LFA

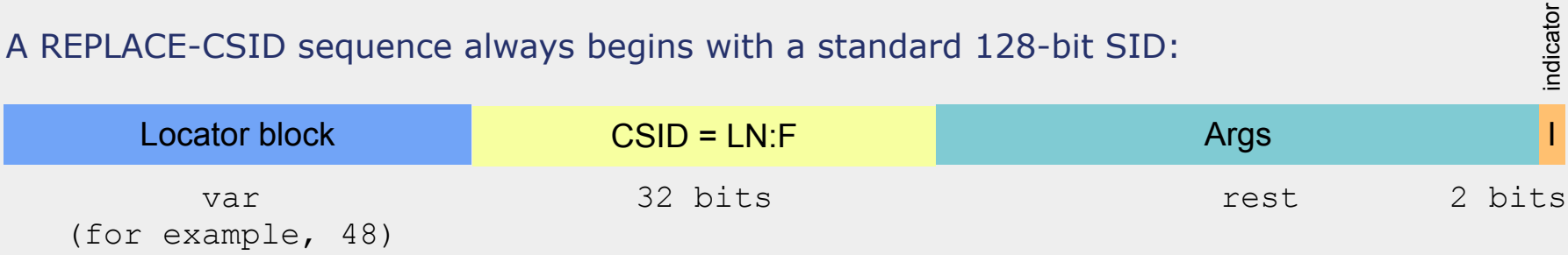
REPLACE-CSID: Huawei approach

If you are too big

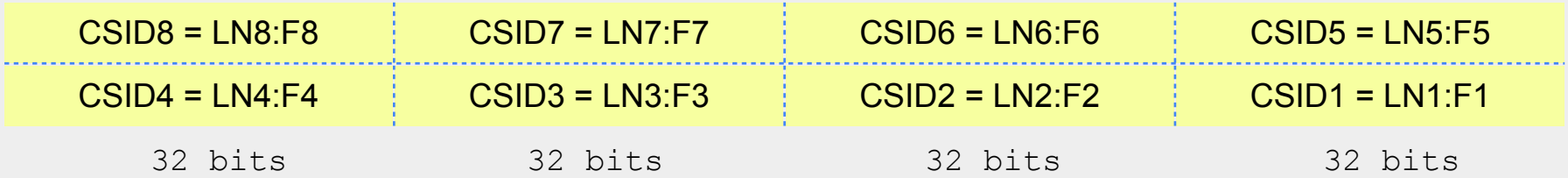
- 6 CSIDs (or uSIDs) may be not enough
  - Then we need an efficient compression for a large number of CSIDs
  - We need scalable algorithms (basically, “everything is the same everywhere”)
- Huawei approach to it: compress data in SRH in a systematic way
  - Generalized SRv6 (G-SRv6)
- There is always exactly one active CSID in use inside the DA
- The next set is stored in the SRH
  - But they are packed in groups of four 32-bit CSIDs within each 128-bit record
- At each hop, the node retrieves the next 32 bits from the SRH and writes them to the DA on top of the current CSID

REPLACE-CSID Structure

A REPLACE-CSID sequence always begins with a standard 128-bit SID:



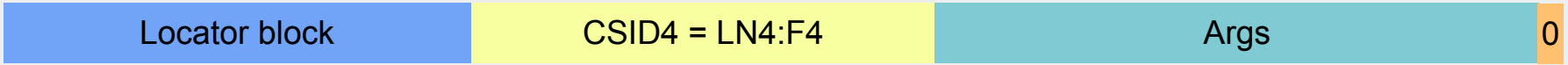
All subsequent CSIDs are stored in the SRH within Segment List entries, each of which is divided into K = 128 / 32 = 4 positions:



REPLACE-CSID traversal

Step 1

DA in the basic header



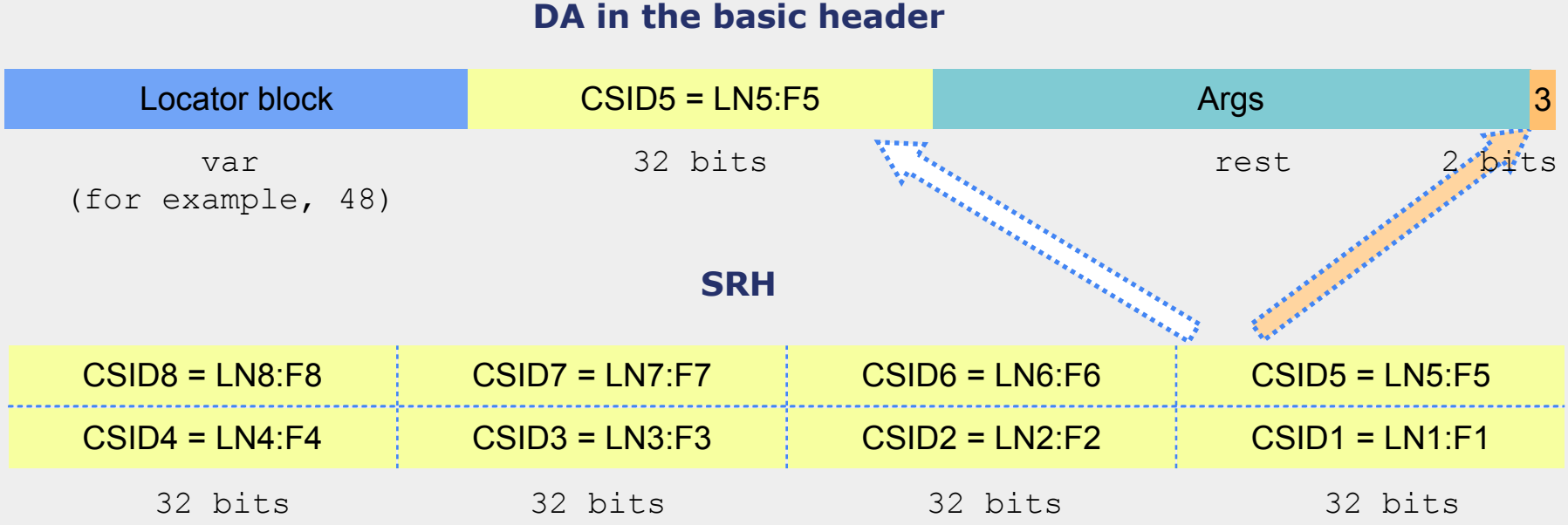
var (for example, 48) | 32 bits | rest | 2 bits

SRH

CSID8 = LN8:F8	CSID7 = LN7:F7	CSID6 = LN6:F6	CSID5 = LN5:F5
CSID4 = LN4:F4	CSID3 = LN3:F3	CSID2 = LN2:F2	CSID1 = LN1:F1
32 bits	32 bits	32 bits	32 bits

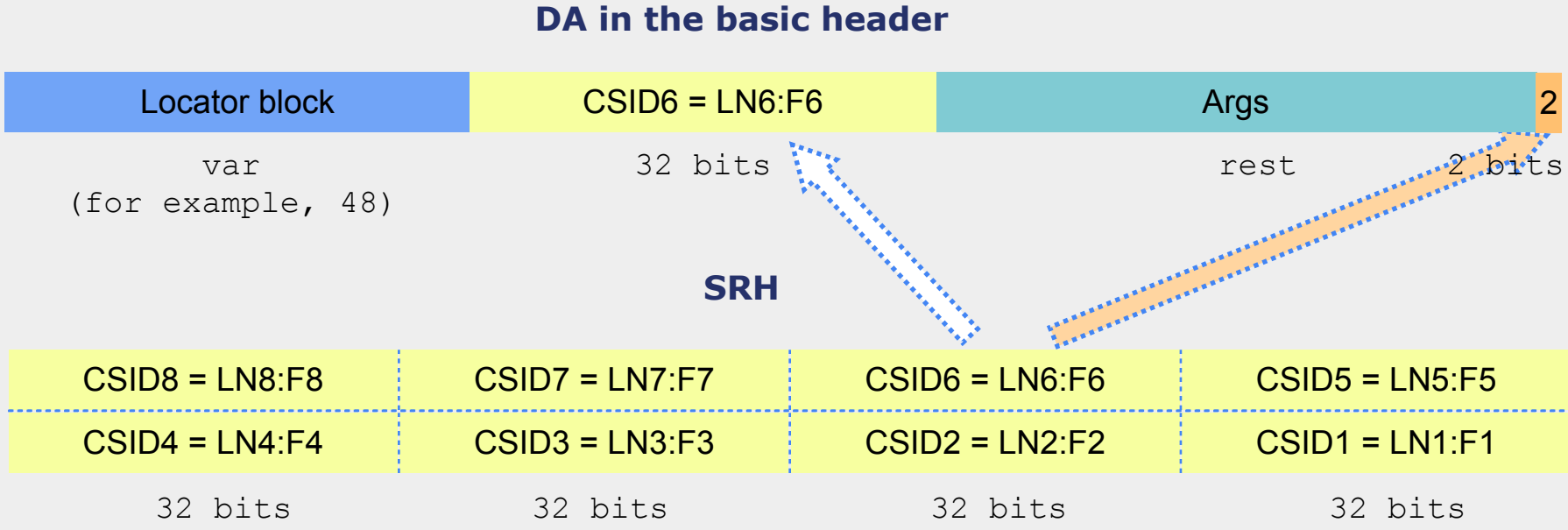
REPLACE-CSID traversal

Step 2: arrived to LN4



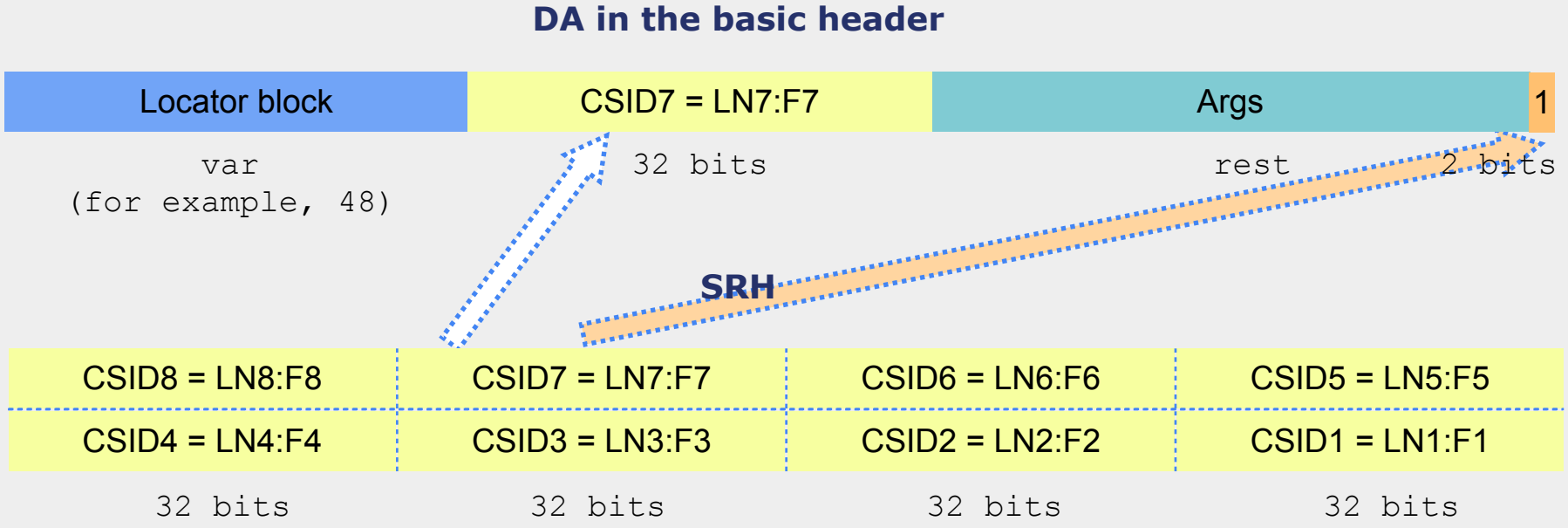
REPLACE-CSID traversal

Step 3: arrived to LN5



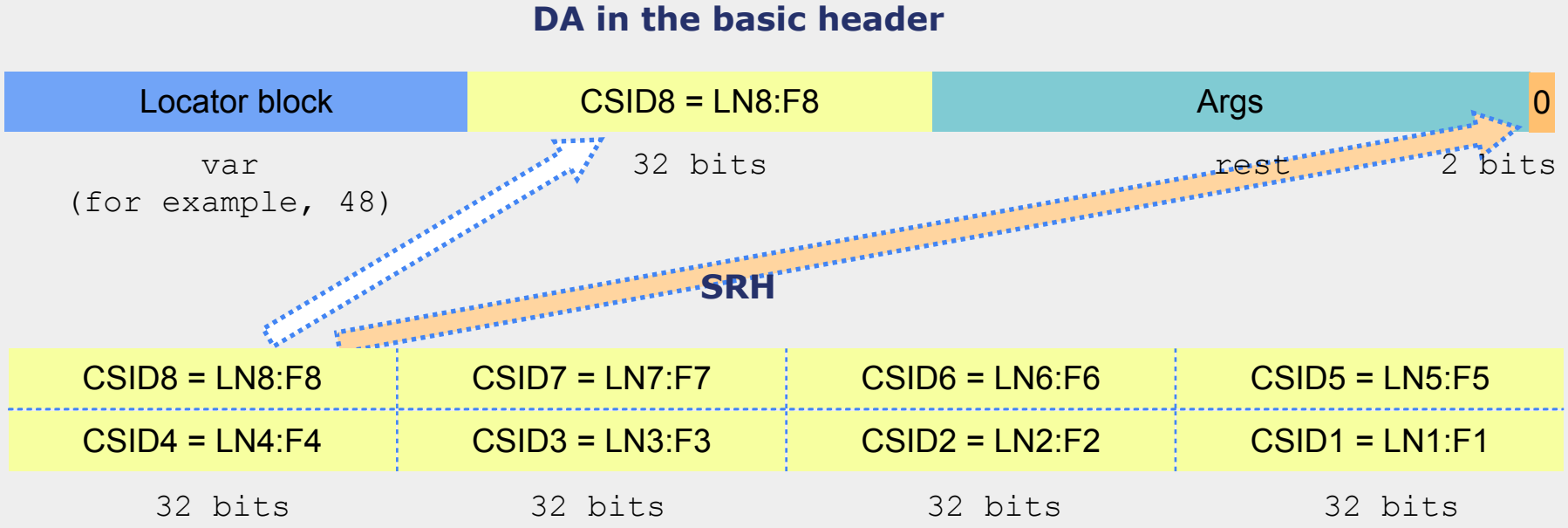
REPLACE-CSID traversal

Step 4: arrived to LN6



REPLACE-CSID traversal

Step 5: arrived to LN7



Security Considerations

Security

SRH is too similar to RH0, we are doomed!

- It is a policy that makes a technology from the set of protocols and header formats
- And it is more than true for SRv6

Security considerations:

- SRv6 operates within the **trust boundary**
- The SID address space **must not be accessible from outside**
  - By internal SRv6-capable nodes as well
- **The control plane** is part of the security perimeter
  - Theoretically, also HMAC signatures for SRH were defined (TLVs in SRH)
  - Currently is supported only by Linux
    - Performance: no performance

Fail-open by design (A.Alston, 2020-2024)

- MPLS, CLNS, BIER are **fail-closed**: a protocol is processed on an interface only if explicitly enabled there
- SRv6 is **fail-open**: it is indistinguishable from IPv6 at L2 (same EtherType) and L3 (same protocol numbers)
  - Any packet that reaches a SID is and must be executed
- The standard requires **a perimeter around SR domain**
  - But offers nothing if the perimeter has **a hole**
- His proposal: a dedicated EtherType for SRv6 (TD-SRv6), imposed only by border routers on packets they encapsulate

Security: fail-open model addressed

IETF answer

- A new EtherType might be an option, but it is too late
  - Too many RFCs to be made obsolete
  - It requires the completely new silicon
- Instead, it was proposed to codify filtering, making the perimeter *mechanical*, not manual
  - One SID block (5f00::/16 or ULA), never advertised outside the domain
  - Infrastructure ACL on every external interface
  - MACsec on transit links
  - Make the edge default-closed
  - Keep untrusted hosts out of the domain
  - Reduce the blast radius: dynamic function allocation, no BGP-LS export to untrusted consumers etc
- What remains: fail-open is a property of the protocol; the mitigations move it into operations

Other Operational Aspects

Operational aspects

The MTU and Hardware budget

- SRv6 is heavy: encapsulation comes with a price
  - Basic outer IPv6 header: 40 bytes
  - SRH base: 8 bytes
  - Each SID in the segment list: 16 bytes
- Network-wide MTU planning is critical
  - Jumbo frames must be configured and aligned consistently
- Hardware limitations (The ASIC reality)
  - ASICs have a limited header parsing depth
  - If the parser cannot scan all the external header, the packet is recirculated or dropped
  - Maximum SID Depth (MSD) parameters are physical hardware limits, not just recommendations
    - They are in those Sub-TLV for a reason

Operational aspects

OAM again: O-bit

- O-flag = bit 2 of the SRH Flags octet
- Every segment endpoint (i.e. node executing its SID) punts a **timestamped copy** of the packet to its OAM process and forwards the original unchanged
  - The copy is usually truncated
  - These copies are exported out-of-band (telemetry) to a collector, which correlates them by packet
- What this gives that classic IP OAM cannot
  - path verification
  - one-way delay and jitter (provided you have PTP-synced clocks)
  - loss localisation
  - no dependency on ICMP
- Complements traditional pings and traceroutes, not replaces them
- **Unavailable on SRH-less uSID paths**

Operational aspects

Even more OAM: RFC 9259

- ICMPv6 Echo to a SID
  - End node processes ICMPv6 as an allowed upper-layer header
- Add an SRH to the probe and the reply verifies an exact segment list, not just reachability
- Traceroute by hop-limit expiry, segment by segment; ICMPv6 errors carry the SRH, so the source sees which segment failed
- End.OP / End.OTP existed in the drafts and were dropped
  - O-flag replaced them

ECMP Support?

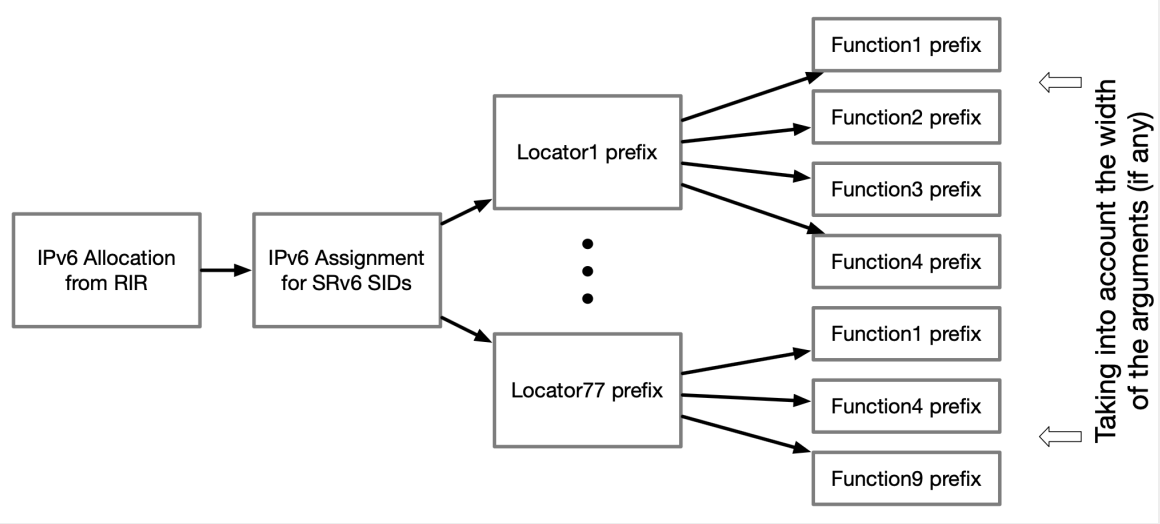
IPv6 Flow Label becomes load-bearing

- 1998, RFC 2460: 20-bit Flow Label "for flow-specific handling”
  - semantics left undefined
  - and we used to it
- 2011, RFC 6437/6438: redefined as a pseudo-random per-flow value
  - set by the source, immutable in transit
  - nobody cared
- **SRv6 turns Flow Label into the only entropy a transit node has**
  - Basic IPv6 header + SRH are too deep to scan through!
  - *.Encaps computes the outer Flow Label from the inner 5-tuple
  - *.Insert keeps an old value
  - Similar to the Entropy Label in MPLS
  - **Necessary for ECMP**

SID are IPv6 addresses

Address plan (for classic SIDs)

- SIDs are **valid IPv6 addresses**
  - **You can choose from your GUA, ULA or 5f00::/16**
  - But they also have to be **aggregated**
- We have to allocate a special **common SR domain prefix** in our allocation



SRv6 is a design, not a feature

SRv6 is not "just enable it on those three routers"

- it is a network-wide address plan
- a trust boundary
- and a forwarding model

SRv6 is a design, not a feature

Planning is mandatory, before your first `locator` line

- Domain scope (which nodes are in, which are out, where the border is)
- Address plan (SID block, locator per node, CSID format vs classic)
- Router capabilities (all those MSD limitations)
- Size of the domain (headends must not form lists longer than min(MSD) along the path)
- Functions and roles (End/End.X everywhere, End.D* on PEs, USD on P?)
- Protection (will do TI-LFA? insert or encap? PSP/USD flavors accordingly)
- Transport prerequisites (IGP support — see Juniper+SRv6+OSPFv3 above, MTU budget, ECMP realisation)

SRv6 is a design, not a feature

A security model must exist

- The domain is fail-open
- Infrastructure ACL must be on every external interface
- Perimeter completeness is a compliance check, not a one-off action
- MACsec on transit links, ICMP policy at the border, decision on whether hosts are inside the domain

Questions?
Comments?
Applause?

asemenyaka@ripe.net