

# Design Experience of an SRv6 uSID Data Center

## SRv6 uSID DC Landscape

Gyan Mishra

“IT Technologist & Innovation Specialist”

Associate Fellow –Network Design

April 9<sup>th</sup> 2024

**Demo Time 1 of 9**

# Demo time!

All Demo's @ YouTube Channel SRv6 uSID DC:

[https://youtube.com/@SRv6\\_uSID\\_DC](https://youtube.com/@SRv6_uSID_DC)

<https://github.com/segmentrouting/srv6-labs> **Directory:** "3-srv6-dc-case-studies"

- ❑ Host-to-Host multi-pod across the metro
- ❑ Policy programmed from Linux Kernel & VPP host

**MPLS WC 2024**

**SRv6 uSID DC Series**

## Use-case 1: SRv6 uSID BGP-Only DC & Single AS DC

- IPv6 DC ⇔ **SRv6 uSID Core** ⇔ IPv6 DC (1a) ⇔ **YouTube Video #1 (1 of 9) Host Networking**
- **SRv6 uSID DC** ⇔ IPv6 Core ⇔ **SRv6 uSID DC** (1b) ⇔ **YouTube Video #2 (2 of 9) Migration**
- **SRv6 uSID uSID** End-to-End (1c) ⇔ **YouTube Video #3 (3 of 9) uSID E2E**

## Use-case 2: SRv6 uSID w/ Multi POD Fabrics

- IPv6 DC ⇔ **SRv6 uSID Core** ⇔ IPv6 DC (2a) ⇔ **YouTube Video #4 (4 of 9) Host Networking**
- **SRv6 uSID DC** ⇔ IPv6 Core ⇔ **SRv6 uSID DC** (2b) ⇔ **YouTube Video #5 (5 of 9) Migration**
- **SRv6 uSID** End-to-End (2c) ⇔ **YouTube Video #6 (6 of 9) uSID E2E**

## Use-case 3: SRv6 uSID w/ Multi POD/Domain Fabrics

- IPv6 DC ⇔ **SRv6 uSID Core** ⇔ IPv6 DC (3a) ⇔ **YouTube Video #7 (7 of 9) Host Networking**
- **SRv6 uSID DC** ⇔ IPv6 Core ⇔ **SRv6 uSID DC** (3b) ⇔ **YouTube Video #8 (8 of 9) Migration**
- **SRv6 uSID** End-to-End (3c) ⇔ **YouTube Video #9 (9 of 9) uSID E2E**



---

## Demo time!

All Demo's @ YouTube Channel SRv6 uSID DC:

<https://github.com/segmentrouting/srv6-labs>

[https://youtube.com/@SRv6\\_uSID\\_DC](https://youtube.com/@SRv6_uSID_DC)

### Use-case 1: SRv6 uSID BGP-Only DC & Single AS DC

**This Deck will be posted to  
GitHub:  
Directory:"3-srv6-dc-case-studies"**

- ☐ Host-to-Host multi-pod across the metro
- ☐ Policy programmed from Linux Kernel & VPP host
- ☐ Policy programmed from Router-In-Container (XRD)



---

# Real World Use-cases

## #1 IPv6 Host Based Networking



---

# #1 IPv6 Host Based Networking

- **Traffic Engineering and Carrier Grade** features are not a requirement in the Data Center.
- Operators can use white box switches or disaggregated hardware and software with Vanilla IPv6 Only DC fabric blindly passing the SRv6 uSID packets. Massive bandwidth **where Multi Petabits** of fiber can be thrown at the DC fabric, with the focus on **High Bandwidth** packet pushing with **Ultra simplified fabric**.
- **Steering** is initiated from the Data Center host attachment using IGP shortest path leaving the entire fabric 100% vanilla IPv6.



# SILOED Networking ⇔ Complexity Tax

Each siloed network Domain has its own Hardware, Software, SDN Stack, Operations & Automation Ecosystem

## IPv4 Transport

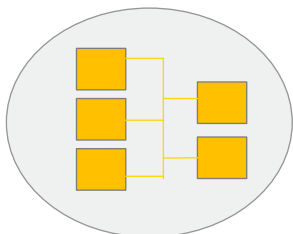
OVERLAY/SEGMENTATION

PRIVATE IP SCALE

TRAFFIC ENGINEERING

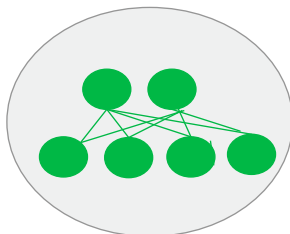


VXLAN / GENEVE



DC OVERLAYS

IPv4



DC FABRICS  
(UNDERLAY)

## MPLS Transport

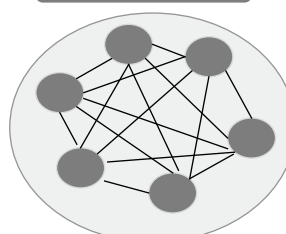
NOT WIDELY ADOPTED IN  
DATA CENTER

LABEL SUMMARIZATION

TRAFFIC ENGINEERING

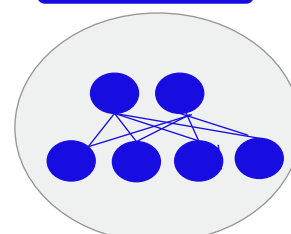


MPLS



METRO  
WAN FABRICS

IPv4, IPv6



EDGE PEERING  
FABRICS



# SRv6 uSID ⇔ Simplicity, Functionality, Ultra Scale

A Common End-to-End Forwarding Architecture Enables Common HW, SW, SDN, Ops ⇔ Massive Scale

## SRv6 uSID

OVERLAY/SEGMENTATION



COMPATIBILITY WITH BROWN FIELD FABRICS



IPv6 UNDERLAY SCALE



SERVICE INSERTION / CHAINING



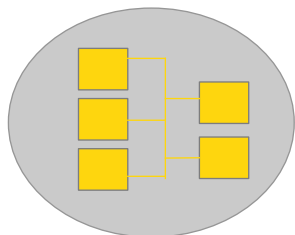
TRAFFIC ENGINEERING



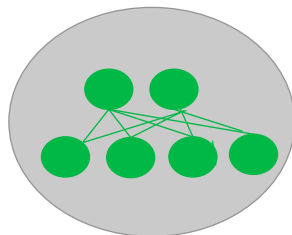
COMMON HARDWARE, SOFTWARE, SDN, Ops



SRv6 uSID ⇔ Translates into Ultra Massive Scale

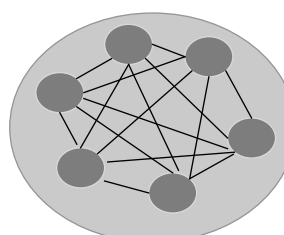


DC OVERLAYS

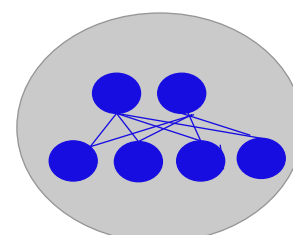


UNDERLAY  
DC FABRICS

“End-to-End  
Inter-Domain  
Routing via  
SRv6 uSID”



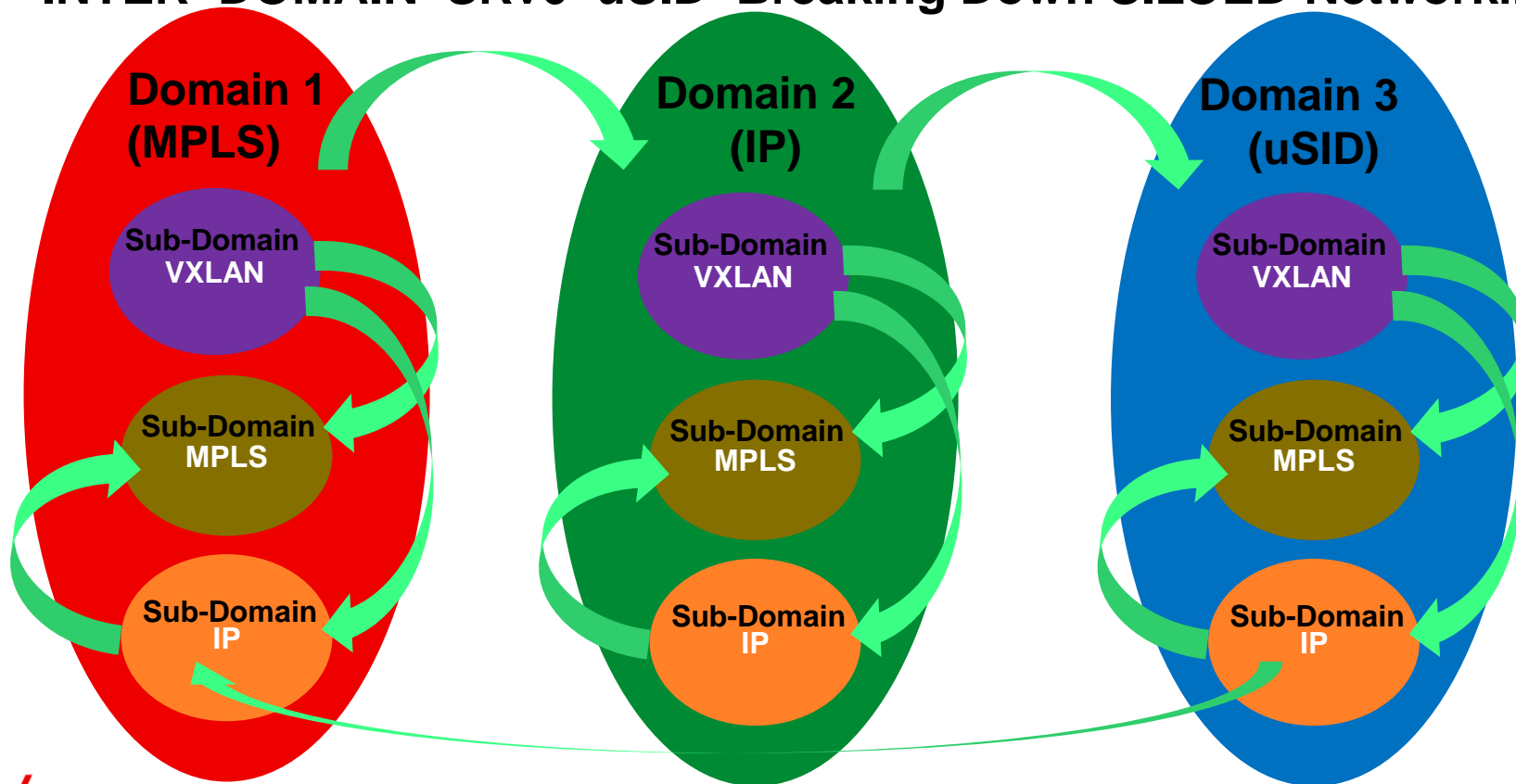
METRO  
WAN FABRICS



EDGE PEERING  
FABRICS

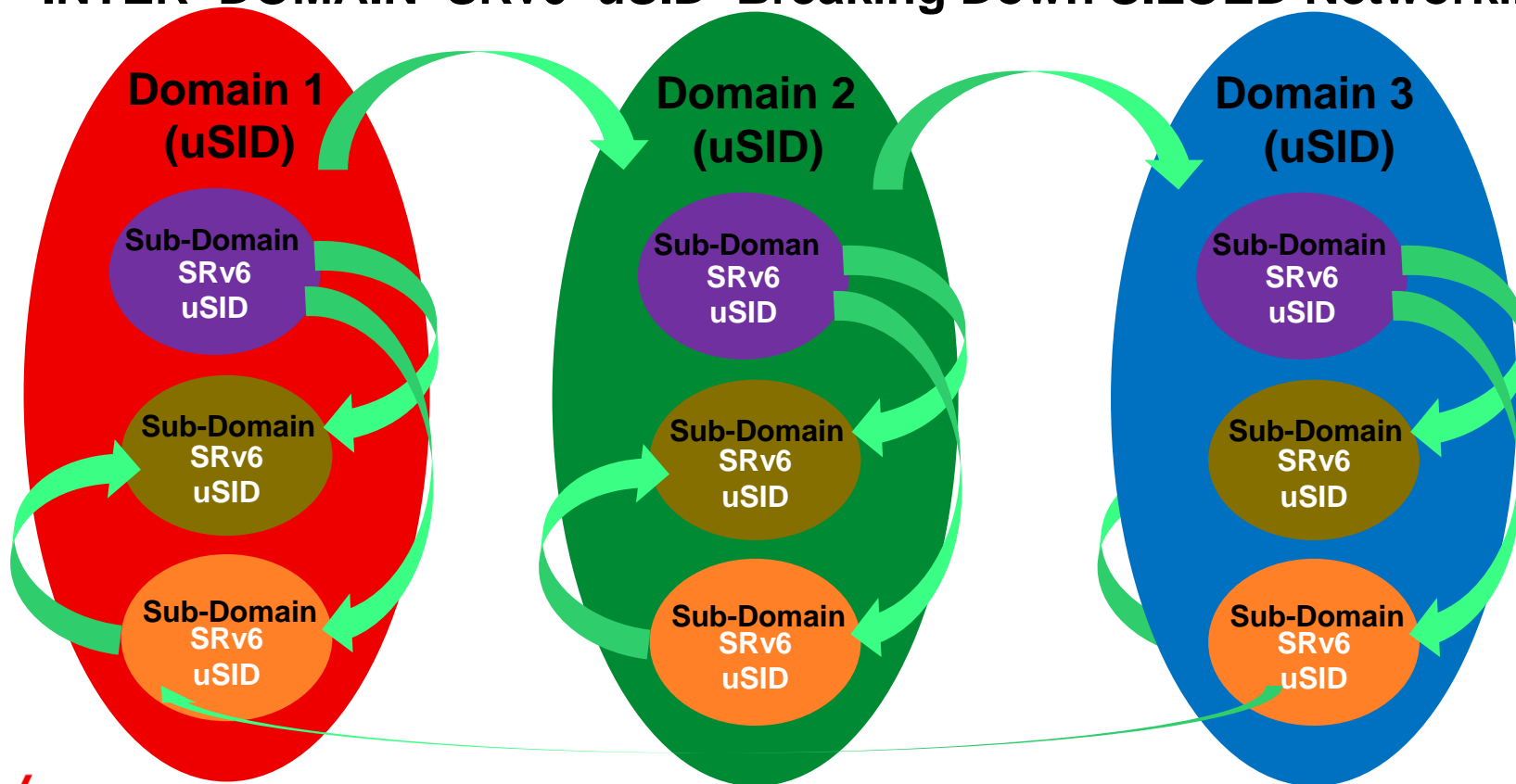


# INTER DOMAIN SRv6 uSID Breaking Down SILOED Networking

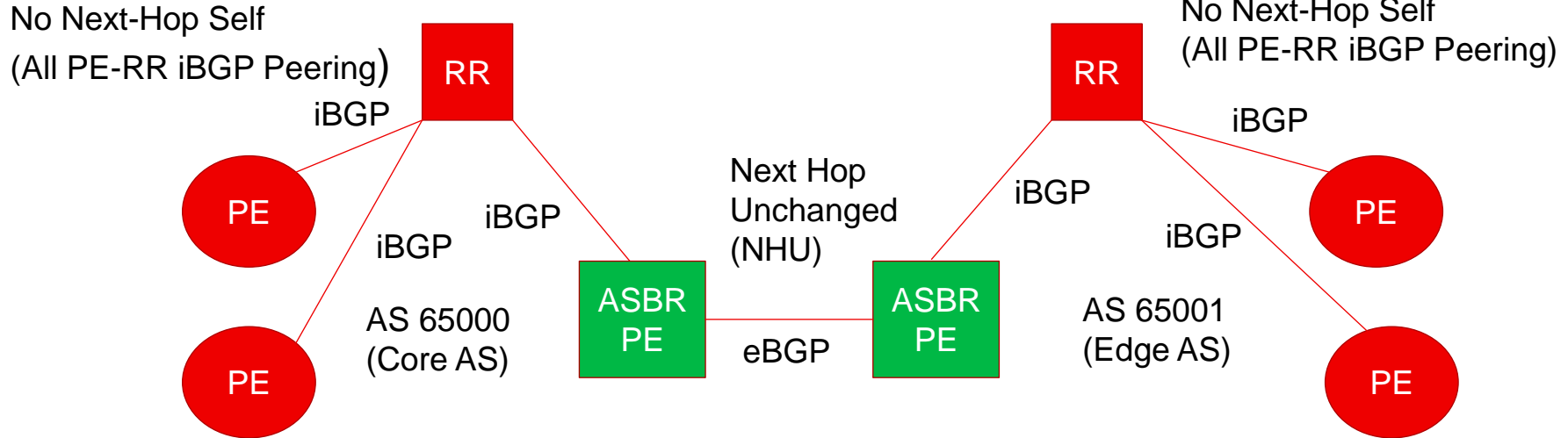




# INTER DOMAIN SRv6 uSID Breaking Down SILOED Networking



# INTER DOMAIN SRv6 uSID



## Rule for Inter Domain Peering

- iBGP -No Next-Hop Self
- eBGP –Next Hop Unchanged

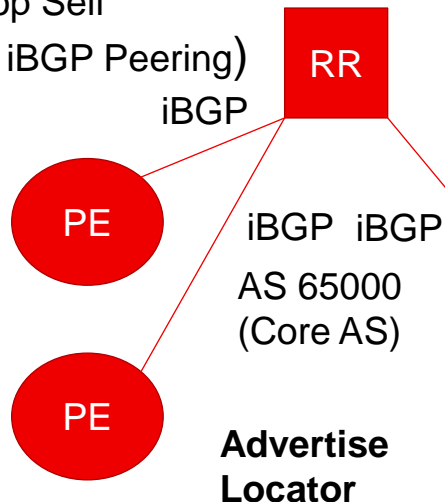
(Requirement to Preserve L2 VPN & L3 VPN Service SID across INTER-AS Boundary)



# INTER DOMAIN SRv6 uSID ROUTING SIMPLICITY

No Next-Hop Self

(All PE-RR iBGP Peering)



**Advertise  
Locator**

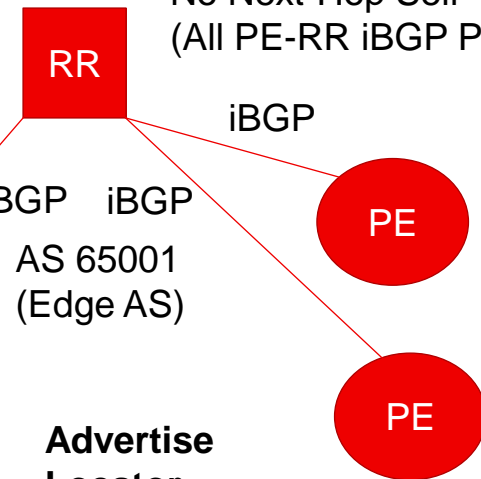
Next Hop  
Unchanged  
(NHU)

eBGP

**ASBR  
PE**

**ASBR  
PE**

No Next-Hop Self  
(All PE-RR iBGP Peering)



**Advertise  
Locator**

## Rule for Inter Domain Peering

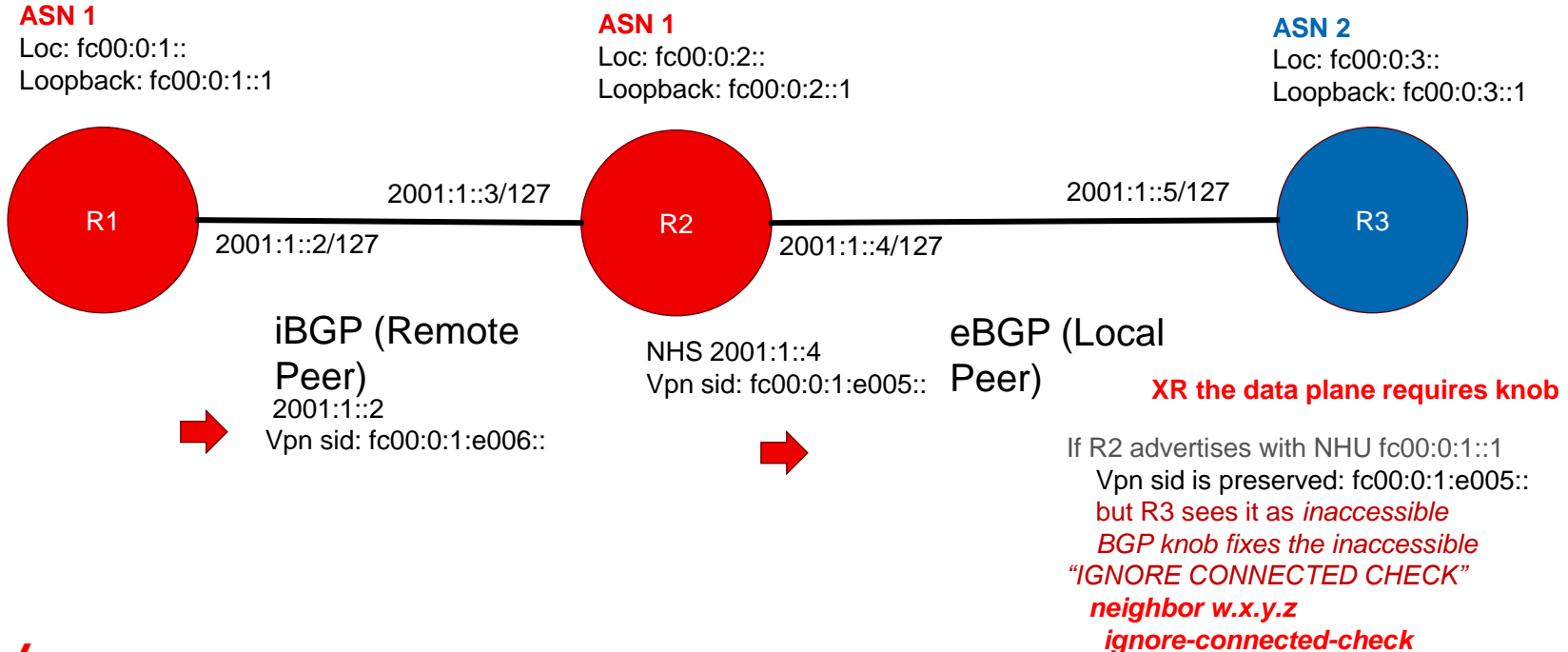
- Locator Reachability (That's it!!)

This allows host endpoints to provide static steering capabilities without PCE across any SR Algo cross domain



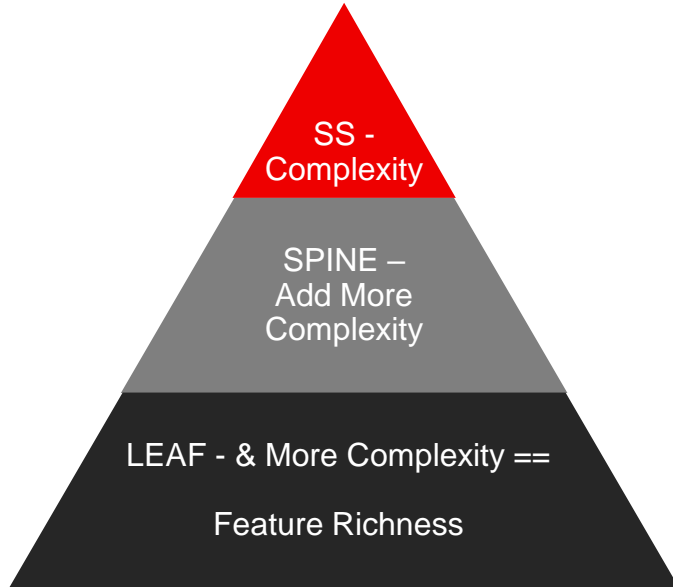
# INTER DOMAIN SRv6 uSID

## eBGP direct peering (NHU) – (Remote PE)



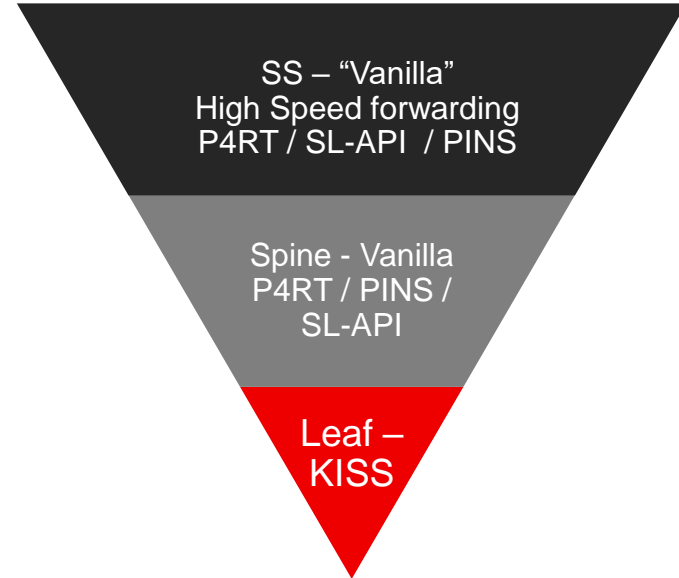
# SRv6 uSID Design ⇔ “Top ⇔ Down” & “Bottom ⇔ Up” Approach

## Top ⇔ Down



Traditional mindset has been for feature richness & complexity across the Data Center Fabric

## Bottom ⇔ UP



SRv6 KISS ⇔ “Keep it Simple & Strategic” approach ⇔ Focus on High speed forwarding plane packet pushing throughput



---

# SRv6 uSID Host Based Networking Traffic Engineering

## Options for Host Based Networking

- ☐ eBPF/Cilium (Cilium BGP control plane) CNI or Standalone (Option-1)
  - ☐ Native Linux Kernel (FRR BGP control plane) –Host Routing with Native Kernel (Option-2)
  - ☐ FD.io VPP (FRR BGP control plane) -Host Routing via VPP (Option-3)
  - ☐ Router-in-container (Control Plane & Data Plane) (xRD, SONiC, Nokia, Juniper cRPD) CNF (Option-4)
- 
- ☐ Options are listed in order of desirability by operators
  - ☐ Next few slides we will go into each option in detail



---

## Option #1 eBPF/Cilium & SRv6 uSID TE Capabilities

- ❑ CNI Connects to global table ⇔ linkage of host fabric to DC fabric
- ❑ CNI TE is Manual/Static today ⇔ Future Roadmap for Dynamic
- ❑ CNI Provides VPN overlay - Workload Container, VM, CNF, VNF

### Details

- ❑ Data plane programming
- ❑ Cilium used for BGP control plane advertisement
- ❑ Cilium is one of the most popular CNI's to date and eBPF with its origins in the Linux kernel with its rich policy features & programmability provides seamless integration to compute nodes making it a powerful win-win for developers
- ❑ eBPF bypasses Linux kernel for policy processing & has direct access to NIC



---

## Option #2 Native Linux Kernel & SRv6 uSID TE Capabilities

- ❑ Connects to global table ⇔ linkage of host fabric to DC fabric
- ❑ TE Capabilities via Linux “iproute 2” support for SRv6 uSID
- ❑ Host VPN overlay - Workload Container, VM, CNF, VNF for VRF attached workload

### Details

- ❑ Data plane programming
- ❑ FRR BGP for control plane advertisement
- ❑ FRR can program the control plane & via Linux Kernel API call program the data plane FIB entries
- ❑ Alternatively, FRR can program the control plane with hook back to Linux Kernel to program the data plane FIB entries





---

## Option #3 FD.IO VPP (Vector Packet Processing) & SRv6 uSID TE Capabilities

- ❑ VPP Connects to global table ⇔ linkage of host fabric to DC fabric
- ❑ VPP Provides Traffic Engineering capabilities via SRv6 uSID
- ❑ VPP Provides VPN overlay - Workload Container, VM, CNF, VNF

### Details

- ❑ Data plane programming
- ❑ FRR BGP for Control Plane Advertisement
- ❑ VPP Seizes Control of the Linux Hosts NIC
- ❑ Requires Netlink or other method to program VPP FIB
- ❑ VPP (Vector Packet Processing) is a high performance network stack that can support high bandwidth & CPU intensive applications



---

## Option #4 Router-in-Container (RIC) & SRv6 uSID TE Capabilities

- ☐ CNF Connects to global table ⇔ linkage of host fabric to DC fabric
- ☐ CNF Provides Traffic Engineering capabilities via SRv6 uSID
- ☐ CNF Provides VPN overlay - Workload Container, VM, CNF, VNF

### Details

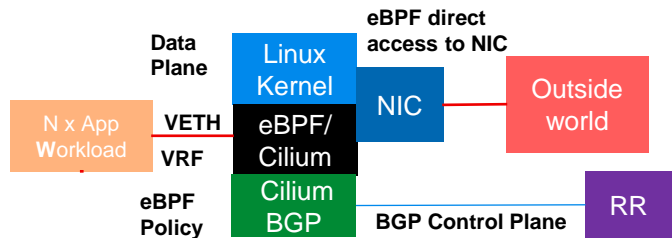
#### Router-in-container options (xRD, SONiC, Nokia, Juniper cRPD)

- ☐ Control plane & Data plane programming
- ☐ Requires Linux bridge stitching between Linux Kernel & CNF
- ☐ Container runs in user space so is beneficial for cases where only certain application requires traffic engineering capabilities
- ☐ User space applications can connect to separate virtual interfaces on router-in-container without any theoretical interface limit thus can support n-app workloads

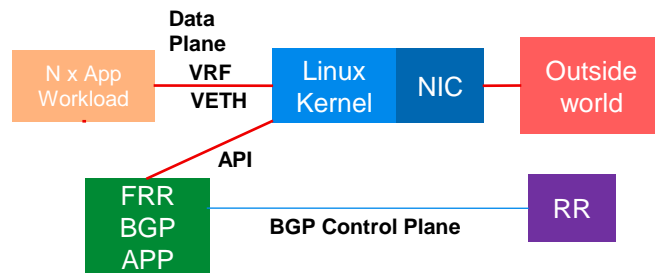


# Host Networking Stacks

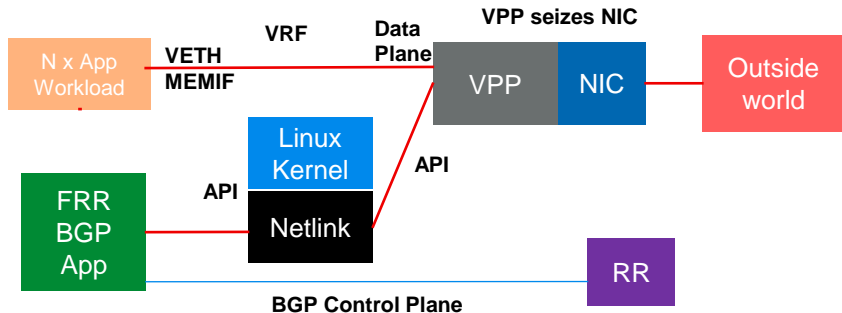
## Option #1 eBPF/Cilium



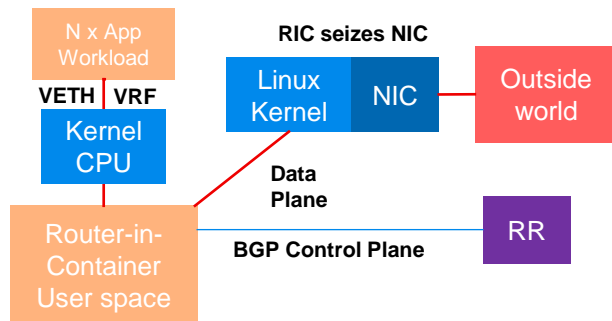
## Option #2 Linux Kernel



## Option #3 FD.io VPP



## Option #4 Router-in-Container (RIC)



# Use-Case 1a: SRv6 uSID E2E w/ IPv6 DC & SRv6 uSID Core

---

Demo time!

All Demo's @ YouTube Channel SRv6 uSID DC:

<https://github.com/segmentrouting/srv6-labs>

[https://youtube.com/@SRv6\\_uSID\\_DC](https://youtube.com/@SRv6_uSID_DC)

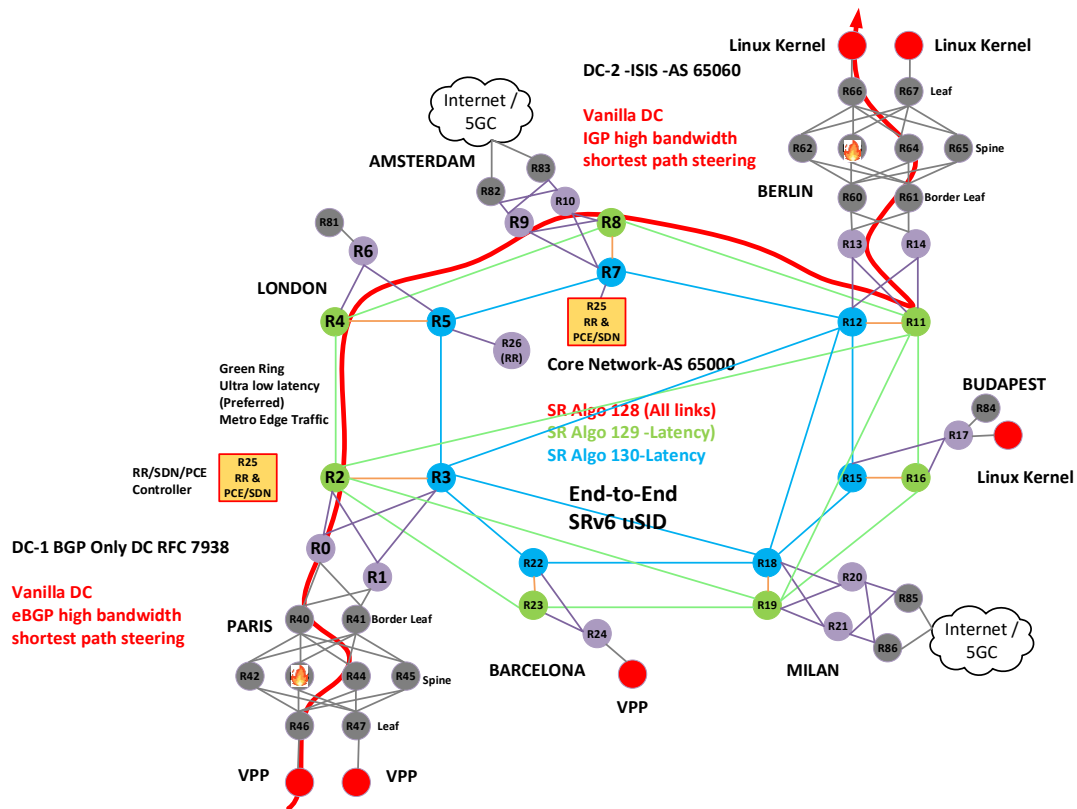
- ☐ Host-to-Host multi-pod across the metro
- ☐ Policy programmed from Linux Kernel & VPP host
- ☐ Policy programmed from Router-In-Container (XRD)

- **Use-case 1: SRv6 uSID BGP-Only DC & Single AS DC**

- **IPv6 DC ⇔ *SRv6 uSID Core* ⇔ IPv6 DC (1a)**



# Use-Case 1a: SRv6 uSID E2E w/ IPv6 DC & SRv6 uSID Core



# Use-Case 1: SRv6 uSID DC Fabric Packet Capture & Screen Scrapes

SRv6 uSID IPv4 payload steer DC-2 Paris ⇌ Core ⇌ DC-1 Berlin using VPP Host attached to DC fabric

## SRv6 uSID IPv4 payload steering policy

```
vpp#sr policy add bsid 40::40 next fc00:0:44:40:4:64:66:e000 encap
```

```
vpp#sr steer l3 10.0.0.66/32 via bsid 40::40
```

```
vpp# show sr policies
```

```
SR policies:
```

```
[0].- BSID: 40::40
```

```
Behavior: Encapsulation
```

```
EncapSrcIP: fc00:0:46:1::3
```

```
Type: Default
```

```
FIB table: 0
```

```
Segment Lists:
```

```
[0].- < fc00:0:44:40:4:64:66:e000 > weight: 1
```

```
vpp# show sr steering-policies
```

```
SR steering policies:
```

```
Traffic      SR policy BSID
```

```
L3 10.0.0.66/32 40::40
```

## IPv4 payload packet capture xrd61-xrd64

```
04:41:56.724814 IP6 fc00:0:46:1::3 > fc00:0:64:66:e000:: IP 10.11.46.2 > 10.0.0.66: ICMP echo request, id 113, seq 463, length 64
```

```
04:41:57.724271 IP6 fc00:0:46:1::3 > fc00:0:64:66:e000:: IP 10.11.46.2 > 10.0.0.66: ICMP echo request, id 113, seq 464, length 64
```

## IPv6 payload packet capture xrd61-xrd64:

```
04:55:37.285381 IP6 fc00:0:46:1::3 > fc00:0:64:66:e000:: IP6 fc00:0:46:2::2 > fc00:0:66::1: ICMP6, echo request, seq 368, length 64
```

```
04:55:38.285012 IP6 fc00:0:46:1::3 > fc00:0:64:66:e000:: IP6 fc00:0:46:2::2 > fc00:0:66::1: ICMP6, echo request, seq 369, length 64
```

## SRv6 uSID IPv6 payload steer:

```
vpp#sr policy add bsid 41::41 next fc00:0:44:40:4:64:66:e000 encap
```

```
vpp# sr steer l3 fc00:0:66::1/128 via bsid 41::41
```

```
vpp# show sr policies
```

```
SR policies:
```

```
[1].- BSID: 94::94
```

```
Behavior: Encapsulation
```

```
EncapSrcIP: fc00:0:46:1::3
```

```
Type: Default
```

```
FIB table: 0
```

```
Segment Lists:
```

```
[1].- < fc00:0:45:41:66:e000:: > weight: 1
```

```
-----  
vpp# show sr steering-policies
```

```
SR steering policies:
```

```
Traffic      SR policy BSID
```

```
L3 fc00:0:66::1/128 41::41
```



## Use-Case 2: SRv6 uSID Inter-DC Packet Capture & Screen Scrapes

SRv6 uSID IPv4 payload steer DC-1 Berlin ⇌ Core ⇌ DC-2 Paris using Linux host attached to DC fabric

### SRv6 uSID IPv6 payload steering policy:

```
root@ubuntu-linux-srv6:sudo ip route add 10.0.0.46/32 encap seg6 mode encap segs fc00:0:64:61:4:44:46:e000 dev ens7
root@ubuntu-linux-srv6:/home/cisco# ip route
default via 192.168.122.1 dev ens8 proto dhcp src 192.168.122.88 metric 100
10.0.0.0/24 via 10.10.66.2 dev ens7 proto static
10.0.0.46 encap seg6 mode encap segs 1 [ fc00:0:64:61:4:44:46:e000 ] dev ens7 scope link----->SRv6 uSID steering programmed IPv4 payload
```

### SRv6 uSID IPv4 payload steer capture DC-2 Paris:

xrd41-xrd44

```
20:30:23.274808 IP6 fc00:0:66:1:5054:2ff:fe41:b107 > fc00:0:44:46:e000::: srcrt (len=2, type=4, segleft=0[|srcrt]
20:30:24.275510 IP6 fc00:0:66:1:5054:2ff:fe41:b107 > fc00:0:44:46:e000::: srcrt (len=2, type=4, segleft=0[|srcrt]
```

### SRv6 uSID IPv6 payload steering policy:

```
root@ubuntu-linux-srv6:sudo ip -6 route add fc00:0:46::1 encap seg6 mode encap segs fc00:0:64:61:4:44:46:e000 dev ens7
root@ubuntu-linux-srv6:/home/cisco# ip -6 route
::1 dev lo proto kernel metric 256 pref medium
fc00:0:46::1 encap seg6 mode encap segs 1 [ fc00:0:64:61:4:44:46:e000 ] dev ens7 metric 1024 pref medium---->SRv6 uSID steering programmed IPv6 payload
```

### SRv6 uSID IPv6 payload steer capture DC-2 Paris:

xrd44-xrd46

```
20:40:32.890109 IP6 fc00:0:66:1:5054:2ff:fe41:b107 > fc00:0:46:e000::: srcrt (len=2, type=4, segleft=0[|srcrt]
20:40:32.890994 IP6 fc00:0:46::1 > fc00:0:66:1:5054:2ff:fe41:b107: ICMP6, parameter problem, code=4, length 176
```



---

# SRv6 uSID features configured in the topology

## SRv6 uSID Features List:

- ☐ Unreachable Prefix Advertisement (UPA) – BGP PIC Trigger
- ☐ SR Policy, ODN with Algo 0, 128, 129, 130 locators, candidate path with weighted SID list (UCMP), Static ERO, Dynamic ERO with Stateful PCE
- ☐ TI-LFA & Microloop Avoidance (uLoop)
- ☐ SR-PM all nodes including router-in-container host attachment
- ☐ PCE all nodes including router-in-container host attachment

