SRv6 Network Programming workshop

Linux Netdev 0x16

Ahmed Abdelsalam
ahabdels@cisco.com

Agenda

Topic	Speaker	Affliation	Duration	Notes
SRv6 update	Ahmed Abdelsalam	Cisco Systems	25	Onsite
ROSE (Research on Open SRv6 Ecosystem) update	Stefano Salsano	University of Rome "Tor Vergata"	10	Remote
Linux Kernel update	Andrea Mayer	University of Rome "Tor Vergata"	25	Onsite
Cilium/eBPF	Daniel Bernier	Bell Canada	20	Remote
SONiC	Reshma Sudarshan	Intel	15	Remote
FRR	Carmine Scarpitta	University of Rome "Tor Vergata"	25	Onsite

SRv6

Record-Speed Standardization and Deployment

Thank you

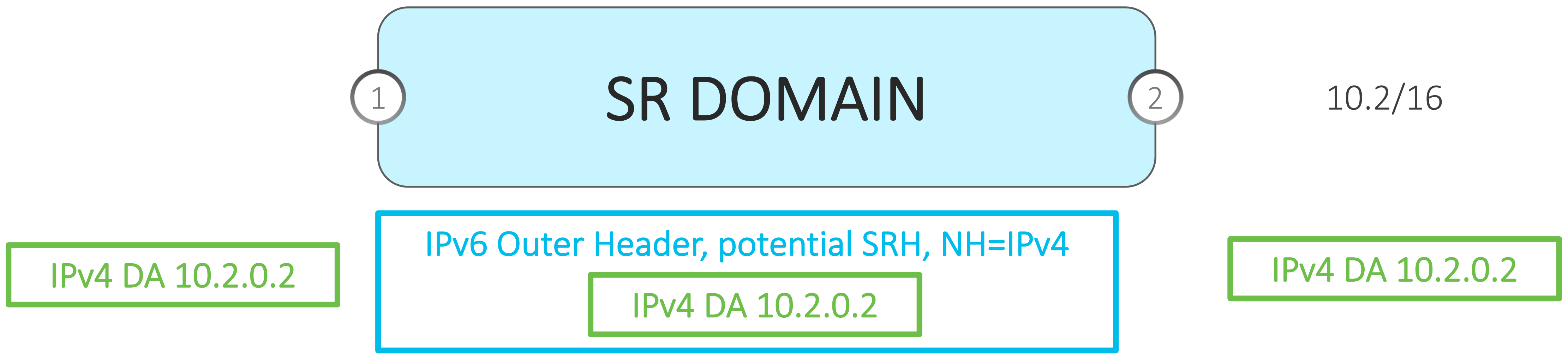
- Lead operators
- EcoSystem Partners
- Academic and Open-Source Partners
- IETF Partners
- Cisco SR Team

segment-routing.net

- SRv6 uSID: CKN, MPLS WC
- SRv6 Ultra-Scale SR Policy: 26 uSID push at linerate: demo
- BGP PIC Edge with SRv6 Summarization: ISIS UPA: demo
- Path Tracing: NANOG85 (Mike Valentine, Fellow at Goldman Sachs), Tutorial

Introduction

Transparent Service



- Customer packet is encapsulated from ingress to egress of the SR Domain
- SRv6 is applied to the OUTER header
- The inner packet is untouched

SRv6 Network Programming - RFC8986

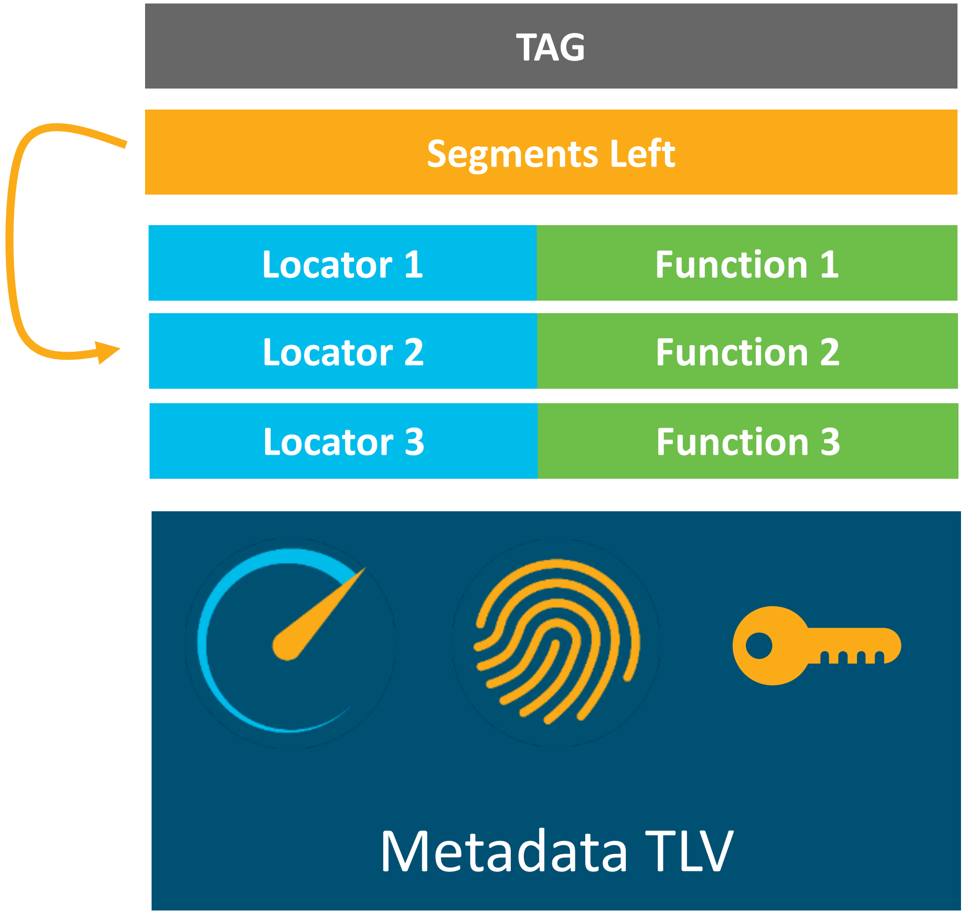
- The End-to-End Policy is encoded as a Network Program
  - The first instruction is in the outer DA
  - The remaining instructions are in the SRH
- An instruction (a SID) may be bound to any behavior
  - TILFA FRR and uLoop Avoidance
  - Traffic Engineering: internal to the domain and across peering links
  - L2/L3 VPN's
  - NFV
  - Any HW custom behavior: P4 on Silicon1
  - Any SW custom behavior: Container orchestrated by Kubernetes
- Powerful Service Creation
  - Any service can be encoded as an ordered list of instructions (Low-latency Slice, VPN, NFV)

Stateless Fabric

- The state (network program) is in the header
- The state is not in the fabric

SR Extension Header

- SRv6 is a native extension of IPv6
  - RFC 8754
  - As foreseen 25 years ago by RFC2460
- SRH contains an ordered list of SID's



Unified Data Plane And Seamless Deployment

- A unified data plane spanning from Application socket to Internet peering though Datacenter, Access, Metro, Core, Peering
- A single data plane natively supported by all nodes – just IP!
- Seamless forwarding through IPv6 transit nodes
- Most use-cases do not need an SRH
  - DA SID contains up to 6 micro-instructions (uSID's)

Host

User Equipment Container

Peering Cell Site

Access Metro

Datacenter

More on SRv6

- Cisco Knowledge Network "SRv6 Standardization Deployed at Scale”
  - https://www.segment-routing.net/tutorials/2021-11-18-CKN-SRv6/

Deployment Status

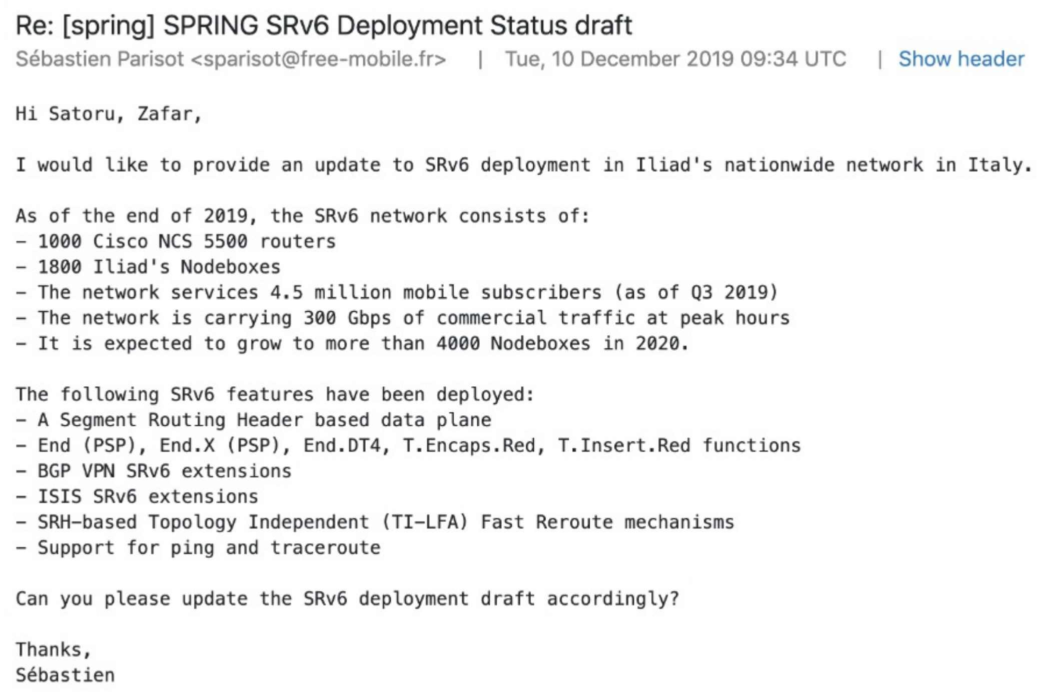
Record-Speed Deployment

- 3 years of commercial deployment (2019-2022)
- ~100M SRv6 subscribers
- ~100 deployments, with ~14 public reports
- Across markets (Web, SP, Enterprise) and geographies (Africa, Asia, EU, US)

SoftBank	Rakuten	Indosat	MTN Uganda	Noia
Iliad	Alibaba	China Telecom	China Bank	Cernet2
Free	Bell Canada	China Unicom	Line	
	NEW			

iliad

- Nationwide deployment in Italy
- 1000 Cisco NCS 5500
- **1800 Iliad Nodeboxes**
  - **Whitebox based on the Linux kernel SRv6 implementation**



Re: [spring] SPRING SRv6 Deployment Status draft
Sébastien Parisot <sparisot@free-mobile.fr> | Tue, 10 December 2019 09:34 UTC | Show header

```
Hi Satoru, Zafar,

I would like to provide an update to SRv6 deployment in Iliad's nationwide network in Italy.

As of the end of 2019, the SRv6 network consists of:
- 1000 Cisco NCS 5500 routers
- 1800 Iliad's Nodeboxes
- The network services 4.5 million mobile subscribers (as of Q3 2019)
- The network is carrying 300 Gbps of commercial traffic at peak hours
- It is expected to grow to more than 4000 Nodeboxes in 2020.

The following SRv6 features have been deployed:
- A Segment Routing Header based data plane
- End (PSP), End.X (PSP), End.DT4, T.Encaps.Red, T.Insert.Red functions
- BGP VPN SRv6 extensions
- ISIS SRv6 extensions
- SRH-based Topology Independent (TI-LFA) Fast Reroute mechanisms
- Support for ping and traceroute

Can you please update the SRv6 deployment draft accordingly?

Thanks,
Sébastien
```

Commitment to SR Lead-Operators

- ✓ Standard-Based Technology
- ✓ Vendor eco-system
- ✓ Open-Source eco-system

Mature Standardization

Much faster standardization than usual

Sign of the SRv6 Industry Endorsement

- Proposed Standard

RFC 8402	SR Architecture
RFC 8754	SRv6 DataPlane
RFC 8986	SRv6 Network Programming
RFC 9252	SRv6 BGP Extension
RFC 9256	SR Policy
RFC 9259	SRv6 OAM

Rich SRv6 Ecosystem

Network Equipment Manufacturers

CISCO HUAWEI NOKIA ciena JUNIPer NETWORKS

ARRCUS NETWORK DIFFERENT ZTE H3C

Merchant Silicon

CISCO Silicon One ™ ©Cisco

BROADCOM. MARVELL™

BAREFOOT NETWORKS intel

Open-Source Applications

Pyroute2 IPTABLES SERA GoBGP exa BGP eBPF cilium

TCPDUMP WIRESHARK SNORT® GO BMP

SR v6

Segment Routing

Open-Source Networking Stacks

.io P4

OPEN Compute Project® SONiC

Smart NIC

intel Mellanox TECHNOLOGIES

Partners

kaloom™ TREND MICRO™ ENEA Qosmos Division

NoviFlow ixia A Keysight Business spirent™ Promise. Assured.

Many Successful Interops

- 2020/04: EANTC: SRv6 interop between Cisco, Huawei, Juniper, Arrcus, Ixia (link)
  - Classic IPv6 nodes as SRv6 transit nodes
  - SRv6-L3VPN for IPv4 and IPv6 services
  - SRv6 TI-LFA FRR link protection with SRH insert
  - SRv6 EVPN for E-Line and EVPN L3VPN services
  - SRv6 TE SR Policy
- 2021/02: NetOne Systems (link)
  - Cisco XR, Cisco NX, Juniper
- 2021/09: EANTC: SRv6 interop between Cisco, Huawei, Juniper, Nokia, Spirent (link)
  - SRv6-Based Global IPv4 and IPv6 services
  - SRv6-L3VPN for IPv4 and IPv6 services
  - SRv6 TI-LFA FRR local SRLG protection with SRH insert
  - SRv6 EVPN for E-Line and EVPN L3VPN services
  - IGP Flex-Algo using TWAMP-measured link delays

SRv6 uSIDs

SRv6 uSID Terminology

- Industry:
  - SRv6 Micro Segment
  - SRv6 uSID
  - Briefly: uSID
- IETF: NEXT-C-SID
  - Briefly: Next
  - IETF document: draft-ietf-spring-srv6-srh-compression-01
  - Training: link

Container of 6 uSID’s

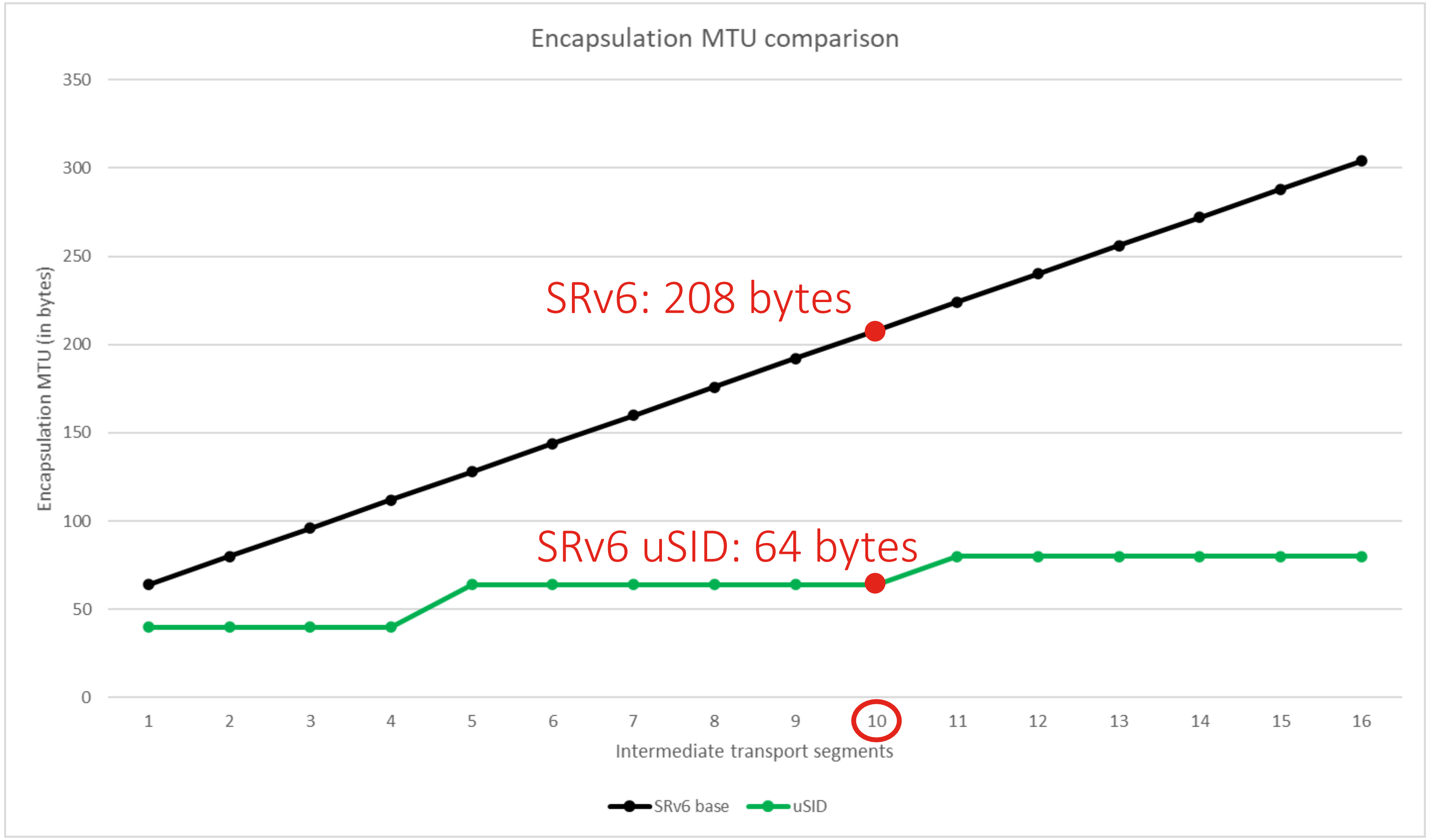
- SRv6 Network Programming (RFC8986)
  - The source encodes any end-to-end program as an ordered list of instructions
  - The first instruction is in the outer DA
  - The remaining instructions are in the SRH
- An instruction is called a SID
- A Container SID may contain up to 6 micro-instructions called uSID’s

FC00:0000	1111	2222	3333	4444	5555	6666
uSID Block	uSID1	uSID2	uSID3	uSID4	uSID5	uSID6

Perfect SRv6 Integration

- uSID reuses SRH (RFC8754) without any change
- uSID strictly applies the SRv6 Network Programming (RFC8986)
- uSID can be bound to any Network Programming Instruction

SRv6 uSID offers the best SRv6 Compression



SRv6 uSID in Deployments

Rakuten – SRv6 uSID in Deployment

- 5G End-to-End Network Slicing based on SRv6 uSID Flex-Algo
  - SRv6 uSIDs are allocated from the ULA address range
  - SRv6 uSID ISIS Flex-Algo: Low-Cost vs Low-Delay
  - SRv6 uSID BGP services
  - SRv6 uSID TILFA
  - Cisco NCS5500 and NCS-540 series
- Innovation in partnership with Cisco
  - BGP PIC Core and Edge with SRv6 Summarization: ISIS UPA (demo)
  - SR BW counters for deterministic and scalable capacity planning and BW guarantee

https://www.segment-routing.net/conferences/MPLS-WC-2022-Amit-Dhamija/

Bell Canada - SRv6 uSID Deployment

- Bell promptly switched from SR-MPLS to SRv6 uSID
- Continued Simplification (remove MPLS dataplane)
- Better Routing Scale: Summarization
- Better HW Scale: linerate 26 uSID push for end-to-end SR Policy
- Seamless Deployment (6 uSID's in DA without SRH)
- End-to-End IP Unified Dataplane from socket to Internet Peering
  - SRv6-TE Policy: topological and service uSID's
- Service Programming
- Reduce network service costs by up to 90%
  footprint by 75%
  power consumption by as much as 66%

https://www.segment-routing.net/conferences/MPLS-WC-2022-Daniel-Voyer/

https://www.segment-routing.net/conferences/MPLS-WC-2022-Daniel-Bernier-Jesper-Eriksson/

SRv6 uSID - Rich Eco-System

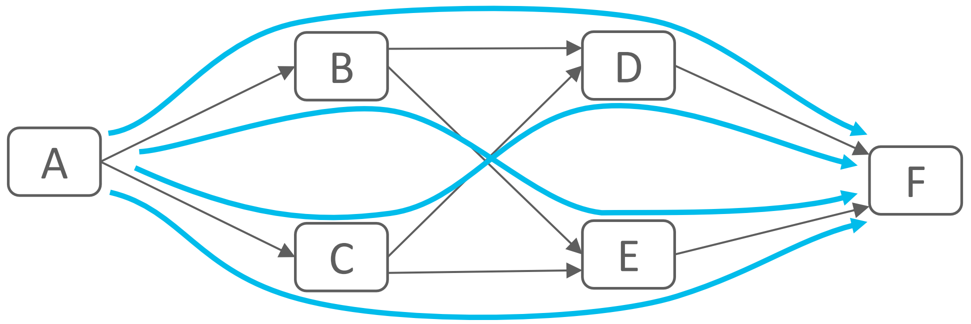
- Cisco, NoviFlow, Arrcus, Nokia, Ciena
- Merchant: Silicon One, Broadcom, Marvell, Barefoot
- Open Source: Linux, FD.io, P4, eBPF, Cillium, SAI, SONIC, FRR

More on uSID

- SRv6 Technology Status and Deployment Update
  - https://www.segment-routing.net/conferences/MPLS-WC-2022-Clarence-Filsfils/

Path Tracing

How did the packet arrive from A to F?

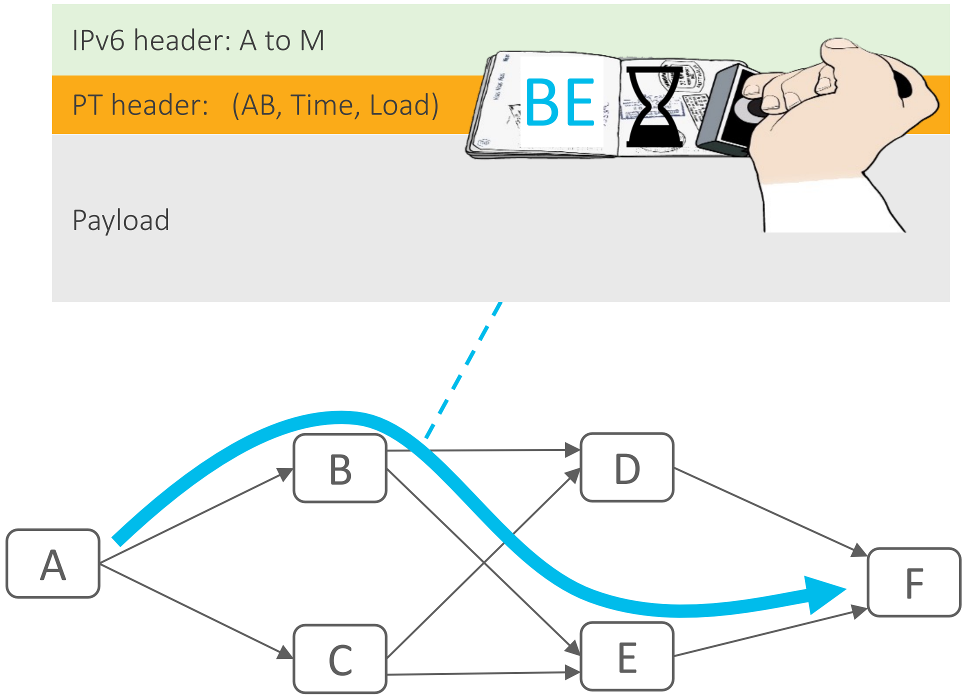


- What exact path was taken by one specific packet?
  - 40-year-old unsolved IP problem
- 4 possible "valid" ECMP paths
  - Packet may have also taken an invalid path (Routing or FIB corruptions)

How does it work?

Stamping Trajectory in PT Header

- Implemented at linerate: Reports **true** packet experience
- Highly compressed for low MTU overhead
  - Only 3 bytes per hop!
- Each transit router records:
  - Outgoing interface ID
  - Timestamp (with 0.06ms accuracy)
  - Load
- Native interworking with legacy nodes
  - Seamless deployment



Ecosystem

- Cisco Shipping in CY22
  - C8000 (Q200), NCS5700 (J2), ASR9000 (LS)
- Rich Eco-System
  - Cisco, Broadcom, Marvell, +others
- Rich Open-Source
  - Linux, FD.io VPP, P4, Wireshark, TCPDUMP

- Path Tracing is being standardized at IETF
  - [Path Tracing in SRv6 networks (ietf.org)]
  - [Path Tracing in SR-MPLS networks (ietf.org)]

Conclusion

Simplicity Always Prevails

~~LDP~~

~~RSVP-TE~~

~~BGP 3107~~

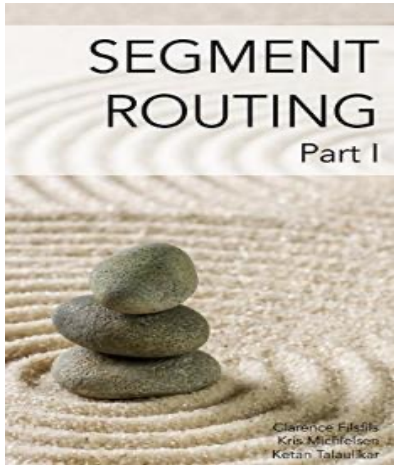
~~MPLS~~

~~UDP/VxLAN~~

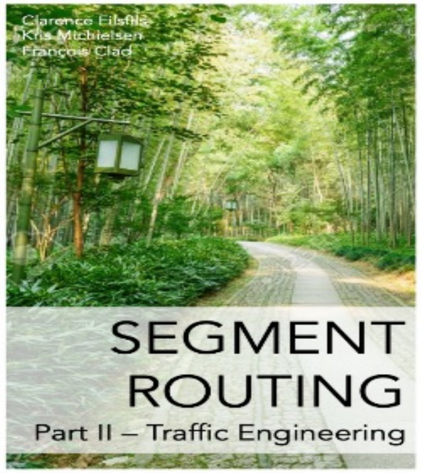
~~NSH~~

Furthermore with more scale and functionality

Stay up-to-date



amzn.com/B01I58LSUO



amazon.com/dp/B07N13RDM9

SRv6 Part III
CY2023

twitter.com/SegmentRouting

segment-routing.net

facebook.com/SegmentRouting/

linkedin.com/groups/8266623

ask-segment-routing@cisco.com

ROSE - Research on Open SRv6 Ecosystem

Stefano Salsano

University of Rome Tor Vergata / CNIT

Linux Netdev 0x16

The ROSE project

ROSE is an “umbrella” project, started in 2017, to develop and maintain an Open Source Ecosystem for SRv6.

The ROSE project has contributed to the standardization of SRv6 in IETF.

Over the years, ROSE has received funding by CISCO, under the CISCO University Research Program.

The ROSE project

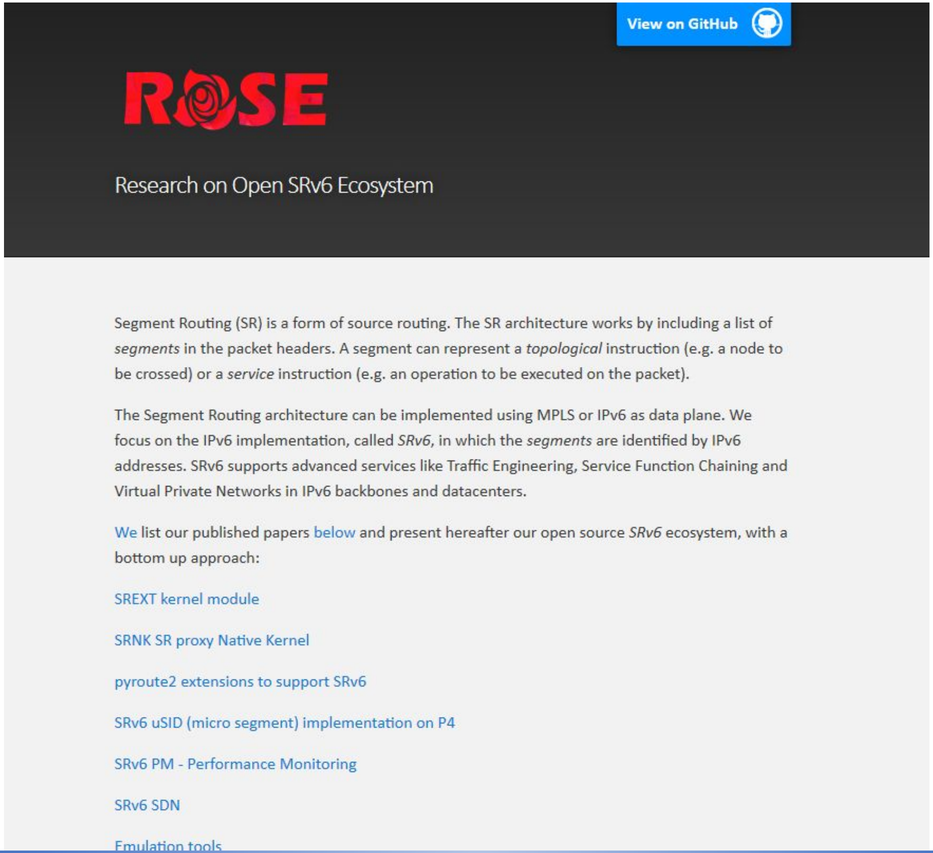
TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

cnit consorzio nazionale interuniversitario per le telecomunicazioni

ROSE includes several sub-projects (10+), related to multiple aspects of the SRv6 technology:

- Data Plane
- Control Plane
- SRv6 host networking stack
- SRv6 integration with applications
- SRv6 integration with Cloud/Data Center Infrastructures

https://netgroup.github.io/rose/



Research on Open SRv6 Ecosystem

Segment Routing (SR) is a form of source routing. The SR architecture works by including a list of *segments* in the packet headers. A segment can represent a *topological* instruction (e.g. a node to be crossed) or a *service* instruction (e.g. an operation to be executed on the packet).

The Segment Routing architecture can be implemented using MPLS or IPv6 as data plane. We focus on the IPv6 implementation, called *SRv6*, in which the *segments* are identified by IPv6 addresses. SRv6 supports advanced services like Traffic Engineering, Service Function Chaining and Virtual Private Networks in IPv6 backbones and datacenters.

We list our published papers below and present hereafter our open source *SRv6* ecosystem, with a bottom up approach:

SREXT kernel module

SRNK SR proxy Native Kernel

pyroute2 extensions to support SRv6

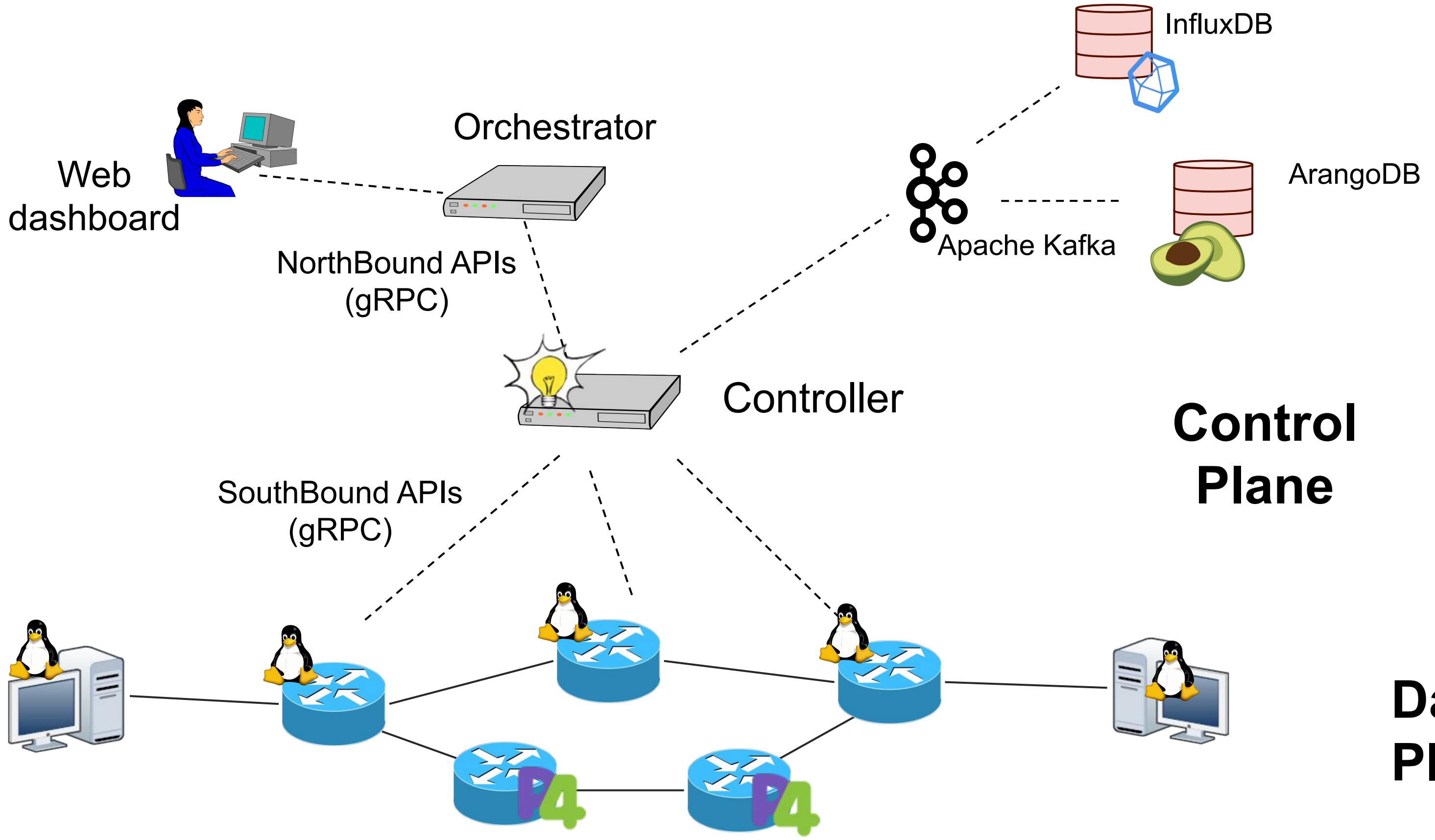
SRv6 uSID (micro segment) implementation on P4

SRv6 PM - Performance Monitoring

SRv6 SDN

Emulation tools

The ROSE ecosystem for SRv6



Big-data Plane

Control Plane

Data Plane

A sample of the sub-projects

- SRv6 **Linux Kernel data plane**
- SRv6 **micro SIDs (P4 and Linux Kernel)**
- **Path tracing** : data plane(s), control plane, algorithms
- **SRv6 VM and Tutorials**
- **k8s-SRv6** : extending Kubernetes to make use of SRv6

SRv6 VM and Tutorials

https://netgroup.github.io/rose/rose-vm.html

A ready-to-go Virtual Machine is available for tutorial and development
It includes an emulated network environment based on Mininet and relies on the Linux kernel for implementing the SRv6 data plane.

Two step-by-step tutorials are included:

- Manual creation of SRv6 tunnels in the Linux SRv6 data plane
- ROSE Control Plane : setting up SRv6 tunnels from the SDN controller

SRv6 VM and Tutorials

https://netgroup.github.io/rose/rose-vm.html

A ready-to-go Virtual Machine is available for tutorial and development
It includes an emulated network environment based on Mininet and relies on the Linux kernel for implementing the SRv6 data plane.

Two step-by-step tutorials are included:

- Manual creation of SRv6 tunnels in the Linux SRv6 data plane
- ROSE Control Plane : setting up SRv6 tunnels from the SDN controller

k8s-SRv6 - Extending Kubernetes with SRv6

https://netgroup.github.io/k8s-srv6/

k8s-SRv6 extends Kubernetes to make use of SRv6

We have extended the Calico-VPP Kubernetes networking plugin with a new SRv6 overlay that supports:

- Encapsulation of both IPv4 and IPv6 pods networking
- Traffic Engineering of the overlay tunnels

Scientific papers

15 scientific papers (list in https://netgroup.github.io/rose/), including this tutorial:

P. L. Ventre, S. Salsano, M. Polverini, A. Cianfrani, A. Abdelsalam, C. Filsfils, P. Camarillo, F. Clad,
"Segment Routing: a Comprehensive Survey of Research Activities, Standardization Efforts and Implementation Results",
IEEE Communications Surveys & Tutorials, Firstquarter 2021

Thank you for your attention!

stefano.salsano@uniroma2.it

SRv6 in the Linux kernel

Andrea Mayer[1]

University of Rome Tor Vergata / CNIT

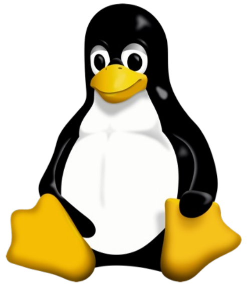
Linux Netdev 0x16

Who I am?

- M.Sc degree in Computer Science and PhD in Electronic Engineering at University of Rome "Tor Vergata";
- Working as researcher engineer at CNIT (National Inter-University Consortium for Telecommunications) on several projects funded by the EU;
- Main interests focus on Linux kernel networking stack, IPv6 Segment Routing (SRv6), Network Function Virtualization (NFV), Virtualization, Software Defined Networking (SDN);
- Developer and contributor to the SRv6 subsystem of the Linux kernel.

Quick agenda

- ❖ **Quick journey** through the evolution of SRv6 in the Linux kernel:
  - ➢ SRv6 initial support (v4.10 ~ v4.18);
  - ➢ **Our contribution** from v5.0 up to now;
  - ➢ Kernel space/User space (i.e.: `iproute2`).
- ❖ What’s next?
- ❖ Conclusions.



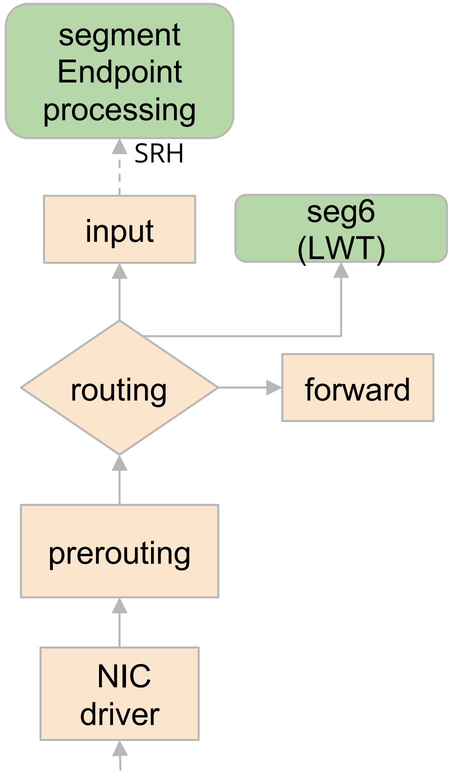
SRv6 support in Linux kernel v4.10 (1/2)

TOR VERGATA UNIVERSITÀ DEGLI STUDI DI ROMA

cnit consorzio nazionale interuniversitario per le telecomunicazioni

4.10

- ❖ Support for SRv6 appears in Linux kernel v4.10;
- ❖ Implement **minimal support** for processing of SR-enabled packets:
  - ➢ Add SRH encapsulation and insertion (`seg6` - LWT);
  - ➢ segment Endpoints processing (***require*** *SRH and DA = local*):
    - ■ Advance the next segment and re-routing;
    - ■ Egress for encap packets: remove of outer IPv6 and SRH;
  - ➢ HMAC support.
- ❖ Endpoint enabled through *per-netns* **sysctl** knobs:
  - ➢ `net.ipv6.conf.default.seg6_enabled`,
    `net.ipv6.conf.<ifname>.seg6_enabled`



SRv6 support in Linux Kernel v4.10 (2/2)

4.10

- `iproute2` extended to:
  - ➢ Support for SRv6 encapsulation (T.Encaps) and insertion (T.Inserts);
  - ➢ Set the source tunnel address;
  - ➢ Handle the HMAC.

- Some examples:
  - ➢ To encapsulate an IPv6 packet into an outer IPv6 + SRH:

```
$ ip ro add 2001:db8::1 encap seg6 mode encap segs fc00::1,fc00::2 dev eth0
```

2001:db8::1 — Dst of packets that will be encap

seg6 — seg6 LWT

mode encap — encap mode

fc00::1,fc00::2 — Segments belonging to the SID List

  - ➢ To set the SRv6 tunnel source address:

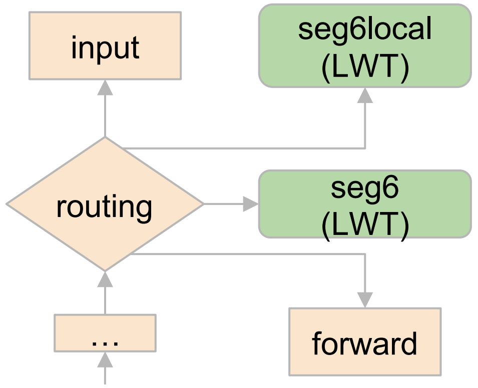
```
$ ip sr tunsrc set 2001:2::1
```

SRv6 support in Linux Kernel v4.14 (1/2)

4.10 4.14

- ❖ SRv6 subsystem has been considerably improved, e.g.,:
  - ➢ Support **SR-encap of IPv4** packets and L2 ethernet frames (*) in IPv6 + SRH;
- ❖ Add support for advanced local segment processing (`seg6local` - LWT):
  - ➢ Implement several "local behaviors" (actions) such as: SRv6 End.X, End.T, End.DX4, etc;
  - ➢ A local behavior can be configured with different (**mandatory**) parameters/attributes;

    Packets to be processed must ***require*** IPv6 DA != local;
    - ■ Some behaviors do not require SRH at all!



(*) Only support ethernet frames with IPv4/IPv6 as L3 proto.

SRv6 support in Linux Kernel v4.14 (2/2)

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

cnit consorzio nazionale interuniversitario per le telecomunicazioni

4.10 4.14

- `iproute2` extended to support advanced local segment processing:
  - set up/destroy local behavior instances;
  - show instantiated behaviors with all the user-provided parameters/attributes.

- Few examples:
  - Instantiate the SRv6 End behavior for the given SID:

$$\texttt{\$ ip -6 ro add } \underbrace{\texttt{2001:db8::1}}_{\text{active SID}} \texttt{ encap } \underbrace{\texttt{seg6local}}_{\text{seg6local LWT}} \texttt{ action } \underbrace{\texttt{End}}_{\text{Behavior to be executed}} \texttt{ dev eth0}$$

  - Instantiate the SRv6 End.T behavior for the given SID:

$$\texttt{\$ ip -6 ro add 2001:db8::1 encap seg6local action End.T } \underbrace{\texttt{table main}}_{\text{attribute \texttt{table}, valorized with \texttt{main}}} \texttt{ dev eth0}$$

SRv6 support in Linux kernel v4.16

- ❖ IPv6 Segment Routing Header (SRH) support for Netfilter:
  - ➢ Provided as a kernel module;
  - ➢ `iptables` CLI integration to set matching rules.
- ❖ It allows matching packets based on SRH;
  - ➢ Supported **match options** include:
    - ■ Next header, Header Extension Length, Segment Left, Last Entry, Tag.
- ❖ It can be combined with other Netfilter extensions to design complex filtering chains and actions:
  - ➢ e.g., implementing SRv6 network packet loss monitoring, delay monitoring, etc.

SRv6 support in Linux Kernel v4.18 (1/2)

4.10 | 4.14 | 4.16 | 4.18

- SRv6 local processing enhanced with the new **End.BPF** action by:
  - A new BPF program type (`BPF_PROG_TYPE_LWT_SEG6LOCAL`);
  - New bpf helpers to encap packets and deal with SR-enabled ones, e.g:
    - `bpf_lwt_push_encap`, `bpf_lwt_seg6_store`, etc.
- End.BPF works like the SRv6 End:
  - SRH must be present;
  - Advance the next segment.
- End.BPF provides an hook for attaching an eBPF program:
  - It can **not** write directly into the packet;
  - Only **some fields** of the SRH (flags, tag and TLVs) can be altered through helper functions.

SRv6 support in Linux Kernel v4.18 (2/2)

4.10 4.14 4.16 4.18

- `iproute2` extended to *load&attach* an eBPF program to End.BPF;
- A file object can contains multiple eBPF programs in different sections:
  - Only one program can be attached to an End.BPF instance.
- For example:
  - *Load&attach* eBPF program "prog1" contained in "foo_obj.o" for the given SID:

```
$ ip -6 route add 2001:db8::6 encap seg6local action End.BPF endpoint \
    object foo-obj.o section prog1 dev eth0
```

File object `foo-obj.o`

eBPF program in section `prog1`

SRv6 support in Linux kernel v5.5 and v5.9

4.10
4.14
4.16
4.18
5.5
5.9

- ❖ Support for local delivery of decap packets in SRv6 End.DT6 (*);
- ❖ The Virtual Routing and Forwarding (VRF) subsystem is an enabling key for implementing new SRv6 behaviors (**);
- ❖ The VRF is extended by supporting the new "**Strict mode**":
  - ➢ It imposes a *one-to-one* mapping between a VRF and the associated Routing Table;
  - ➢ Network-namespace aware;
  - ➢ It can be turned on/off by acting on the "`strict_mode`" **sysctl** knob:
    - ■ `net.vrf.strict_mode` (disabled by default for legacy purposes).

(*) since kernel v5.5, (**) since kernel v5.9

SRv6 support in Linux Kernel v5.11 (1/3)

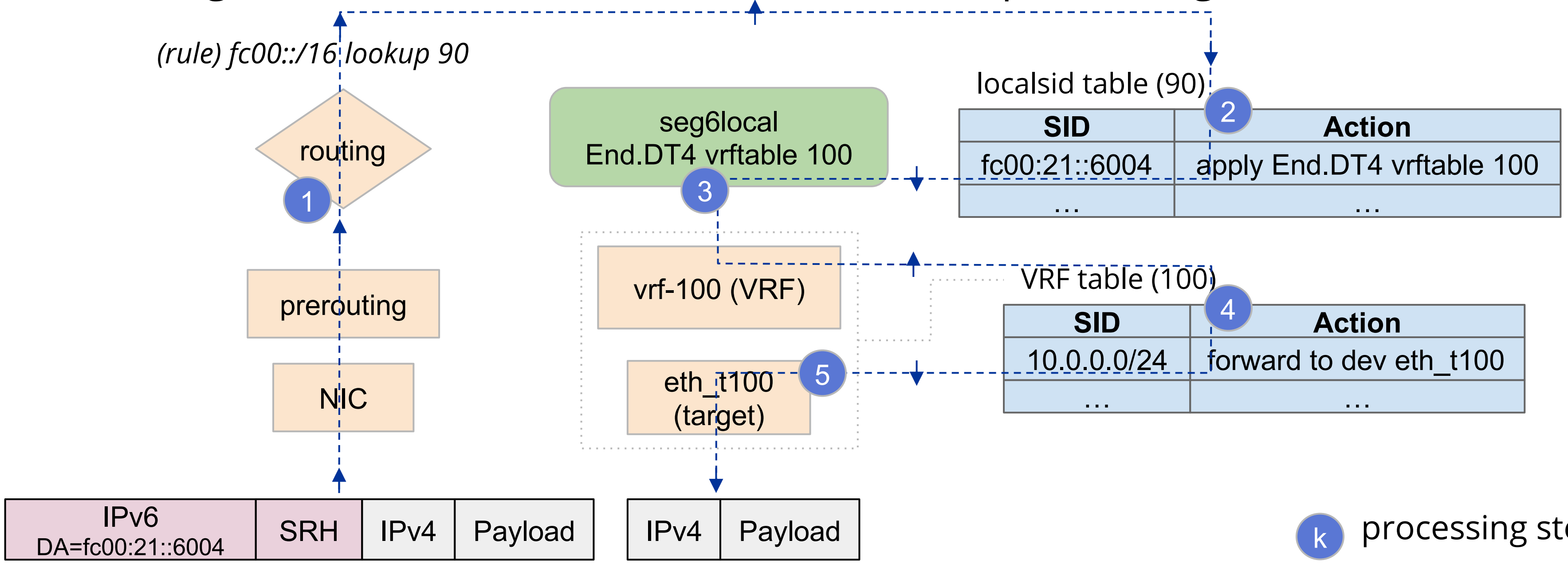
4.10
4.14
4.16
4.18
5.5
5.9
5.11

- ❖ Local processing of SRv6 (`seg6local`) subjected to heavy lifting:
  - ➢ Improved the management of behavior attributes;
  - ➢ Added support for **optional attributes** used by behaviors;
  - ➢ Added callbacks for customizing creation/destruction of behaviors.
- ❖ Add support for SRv6 **End.DT4** behavior:
  - ➢ It decaps inner IPv4 packets and performs lookup into a given Routing Table (RT):
    - ■ Do not strictly require SRH.
  - ➢ It leverages the **VRF** to force the routing lookup into the associated RT:
    - ■ VRF "`strict_mode`" must be turned on!
- ❖ Enhance the SRv6 End.DT6 operating mode:
  - ➢ Legacy mode (providing RT) or using a VRF as in the End.DT4 case.

SRv6 support in Linux Kernel v5.11 (2/3)

4.10
4.14
4.16
4.18
5.5
5.9
5.11

- ❖ A high-level view on SRv6 End.DT4 behavior processing:



localsid table (90)

SID	Action
fc00:21::6004	apply End.DT4 vrftable 100
…	…

VRF table (100)

SID	Action
10.0.0.0/24	forward to dev eth_t100
…	…

k processing step

SRv6 support in Linux Kernel v5.11 (3/3)

4.10
4.14
4.16
4.18
5.5
5.9
5.11

- `iproute2` extended to support both SRv6 End.DT4 and End.DT6 (VRF mode);
- `iproute2` does **not require** any **change** to support optional attributes for SRv6 local behaviors;
- For example, to instantiate an SRv6 End.DT4 behavior for a given SID:

```
$ sysctl -wq net.vrf.strict_mode=1
$ ip link add name vrf-100 type vrf table 100
[...] set the target device of the VRF connecting with the host [...]
$ ip -6 r a 2001:db8::6b encap seg6local action End.DT4 vrftable 100 dev eth0
```

RT associated with VRF vrf-100

SRv6 support in Linux Kernel v5.15 (1/2)

4.10
4.14
4.16
4.18
5.5
5.9
5.11
5.13

❖ Add **counters** support for SRv6 local processing;

❖ For **each** local behavior instance they count:

- ➢ Total number of correctly processed packets;
- ➢ Total amount of traffic (in bytes) correctly processed;
- ➢ Total number of packets **NOT** correctly processed.

❖ Counters are very interesting for:

- ➢ Network monitoring purposes;
- ➢ Checking whether a behavior works as expected or not;
- ➢ Troubleshooting purposes.

❖ Counters **can** be enabled on a behavior instance during the setup phase.

SRv6 support in Linux Kernel v5.15 (2/2)

4.10
4.14
4.16
4.18
5.5
5.9
5.11
5.13

❖ By extending `iproute2`, each SRv6 behavior instance can be configured to make use of counters;

❖ SRv6 counters are supported for any SRv6 local behavior (`seg6local`) as follows:

➢ Add a new SRv6 End behavior instance with the given SID and counters turned on:

```
$ ip -6 route add 2001:db8::1 encap seg6local action End count dev eth0
```

`count` is an optional attribute

➢ Per-behavior counters can be shown by adding "`-s`" to the `iproute2` CLI, e.g.:

```
$ ip -s -6 route show 2001:db8::1
2001:db8::1 encap seg6local action End packets 0 bytes 0 errors 0 dev eth0
```

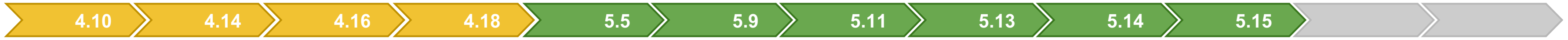
counters (aka statistics)

SRv6 support in Linux kernel v5.14

4.10 › 4.14 › 4.16 › 4.18 › 5.5 › 5.9 › 5.11 › 5.13 › 5.14

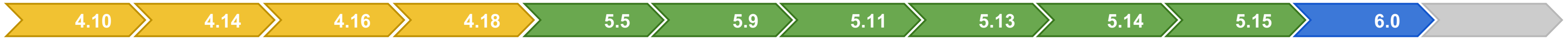
- Add support for **SRv6 End.DT46** behavior:
  - With End.DT4 and End.DT6 is **not** possible to create SRv6 tunnel carrying both IPv4 and IPv6.
- End.DT46 decaps **both** IPv4/IPv6 traffic and routes traffic using a VRF:
  - It reuses the core implementation of End.DT4 and End.DT6 (VRF mode);
  - The VRF "`strict_mode`" must be enabled.
- Performance tests show no degradation in performance when DT46 is used w.r.t. End.DT4/6:
  - End.DT46 greatly **simplifies** the setup and operations of SRv6 VPNs.
- `iproute2` updated to support the new SRv6 End.DT46 behavior:
  - similar CLI and configuration required for setting End.DT4 and End.DT6 (VRF mode).

SRv6 support in Linux kernel v5.15



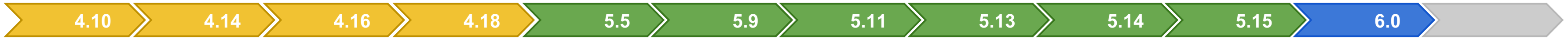
- ❖ Add optional **Netfilter hooks** to SRv6 processing;
- ❖ Netfilter hooks useful to track (*conntrack*) both inner flows and outer flows.
- ❖ By default, Netfilter hooks for SRv6 are disabled:
  - ➢ It can impact on performance when turned on;
  - ➢ **Sysctl** (system-wide) toggle for enabling LWT tunnel netfilter hooks:
    - ■ `net.netfilter.nf_hooks_lwtunnel`
  - ➢ Disabling the `nf_hooks_lwtunnel` requires kernel reboot.

SRv6 support in Linux Kernel v6.0 (1/2)



- Add support for SRv6 **Headend Reduced**:
  - H.Encaps.Red reduced version of H.Encaps.
  - H.L2Encaps.Red reduced version of H.L2Encaps.
- The H.(L2)Encaps.Red **reduces** the length of the SRH by:
  - Excluding the first segment (SID) from the SID List carried by SRH;
  - Pushing the excluded SID directly into the IPv6 DA.
- The H.(L2)Encaps.Red **can avoid** the SRH at all if the SRv6 policy contains only one SID.

SRv6 support in Linux kernel v6.0 (2/2)



- `iproute2` updated to support both H.Encaps.Red and H.L2Encaps.Red;
- Two new *mode* are available to encap `seg6` in `iproute2` CLI:
  - `encap.red` for SRv6 H.Encaps.Red behavior;
  - `l2encap.red` for SRv6 H.L2Encaps behavior.
- Same iproute2 CLI syntax to perform reduced encaps, for example:
  - Perform a reduced encapsulation of an IPv4 packet into an outer IPv6 + SRH

```
$ ip -4 ro a 10.0.0.2 encap seg6 mode encap.red segs fc00::1,fc00::2 dev eth0
```

$\underbrace{\texttt{encap.red}}_{\text{H.Encaps.Red}}$ $\underbrace{\texttt{fc00::1,fc00::2}}_{\text{SID List is transparently reduced by the Linux kernel}}$

SRv6 support in Linux Kernel v6.1-rc1 (1/2)

4.10
4.14
4.16
4.18
5.5
5.9
5.11
5.13
5.14
5.15
6.0
6.1-rc1

- Some SRv6 scenarios may require **large** number of SIDs;
- **Reducing** the **size** of a SID List is useful to:
  - Minimize the impact on MTU; enable SRv6 on legacy HW with limited processing power.
- Kernel v6.1 introduces the **NEXT-C-SID** (aka **uSID**) [1] mechanism for SRv6:
  - Efficient **representation** (compression) of the SID LIst;
    - Several SRv6 segments can be encoded within a single 128-bit SID.
  - NEXT-C-SID mechanism relies on the **“flavors”** framework (RFC 8986):
    - Additional operations that can modify/extend existing behaviors.
- SRv6 **End** behavior is extended with the NEXT-C-SID flavor support.

[1] - https://datatracker.ietf.org/doc/html/draft-ietf-spring-srv6-srh-compression

SRv6 support in Linux Kernel v6.1-rc1 (2/2)

4.10 › 4.14 › 4.16 › 4.18 › 5.5 › 5.9 › 5.11 › 5.13 › 5.14 › 5.15 › 6.0 › 6.1-rc1

- `iproute2` extended to support flavors framework;
- New "`flavors`" attribute to set up NEXT-C-SID **compression** on SRv6 End:
  - Nested ***sub-flavors*** attributes are allowed to further configure the behavior;
- NEXT-C-SID flavor for SRv6 End behavior can be configured using **optional** user-provided *sub-flavors* attributes:
  - `lblen`, i.e.: attribute for Locator-Block length in bits (*> 0 and evenly div by 8*);
  - `nflen`, i.e.: attribute for Locator-Node Function length in bits (*> 0 and evenly div by 8*).
- For example:

  -
    ```
    $ ip -6 ro a 2001:db8::1 encap seg6local action End \
     flavors next-csid lblen 48 nflen 16 dev eth0
    ```

`flavors` attribute accepts *nested attributes*

nested attributes are optionals

What's next?

- ❖ We are still working to:
  - ➢ Introduce new SRv6 features into the kernel;
  - ➢ Cooperate **maintaining** the SRv6 subsystem, e.g. bug fixing, performance tests, etc.
- ❖ (Some of) Upcoming "desidered" features will:
  - ➢ Extend eBPF helper function to support H.Encaps.Red behavior;
  - ➢ Extend the End.X/End.T behaviors to support NEXT-C-SID compression;
  - ➢ Implement the PSP, USP, USD flavors (RFC 8986);
  - ➢ Add counters support to SRv6 Headend behaviors (i.e.: `seg6`);
  - ➢ Set user-provided Traffic Class, Hop Limit in the outer IPv6 header on encap;
  - ➢ Set the tunnel source address (outer IPv6 SA) on a per-behavior basis during encap.
  - ➢ Introduce Path tracing protocol and implementation (leveraging SRv6);
  - ➢ Provide support for SRv6 L2 EVPN based on a new type of virtual network device, i.e.: seg6l2.

Conclusions

- ❖ SRv6 support enhanced considerably across the Linux kernel releases;
  - ➢ We heavily **contributed** to add new features and fix bugs.
- ❖ We extended the VRF and SRv6 subsystems by adding key features, e.g.:
  - ➢ **“Strict mode”** to 1:1 map between a VRF with its RT, used by SRv6 End.DT* behaviors;
  - ➢ **Optional attributes** to make possible complex SRv6 behavior configurations;
  - ➢ Support for built-in **SRv6 counters** made for statistics and troubleshooting purposes;
- ❖ Our work was/is driven by both research and real use-cases needs:
  - ➢ Providing solution for Multi-tenant **IPv4/IPv6 VPNs**, e.g.: End.DT4/6/46;
  - ➢ Avoiding SRH **overhead** whenever possible, e.g.: with reduced encaps;
  - ➢ **SID compression** for supporting SRv6 legacy HW and reducing overhead, i.e.: NEXT-C-SID.
- ❖ *We are very active on SRv6, if you have suggestions or new ideas, please drop us a message!*

Thank you for your attention!

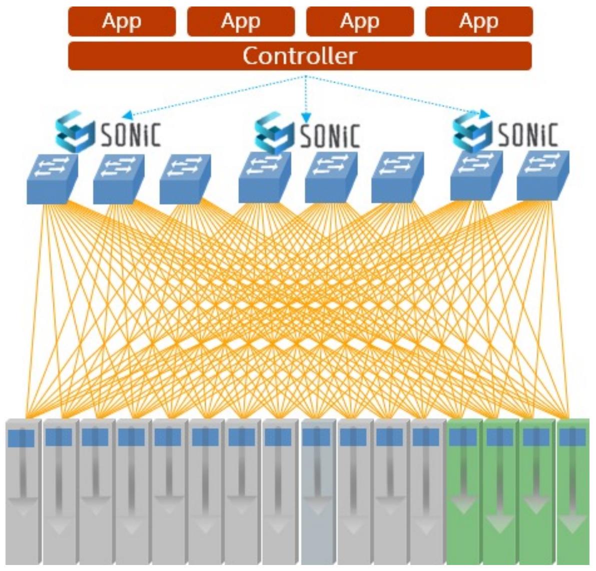
andrea.mayer@uniroma2.it

SRv6 in SONiC
Linux Netdev 0x16

Reshma Sudarshan, Director of Applications Engineering, Intel

SONiC

SONiC is an open source network operating system based on Linux that runs on switches from multiple vendors and ASICs. Microservices architecture breaks monolithic switch software into multiple containerized components that accelerates software evolution



Decouples Hardware & Software

SONiC is built on Switch Abstraction Interface that helps in accelerating hardware innovation

DASH stands for Disaggregated API for SONiC Hosts

SONiC is being extended to Edge devices. DASH community is developing set of APIs and object models describing network services for the cloud, and working with all cloud providers and enterprise hybrid clouds to develop further functionality.

SRv6 benefits to the Operators

Challenges faced by CoSPs today

- Traditional protocols do not give CoSPs the ability to change or modify the behavior of their networks to deliver on service level agreements (SLAs) and engineer the deterministic latency needed for user applications.
- Mobile bearer networks are all independent network domains isolated from each other.
  Applications, however, require packets to traverse these network boundaries.

Solution

Segment routing for IPv6 (SRv6) is an SDN solution for source routing that provides powerful programming ability and a flexible solution for traffic engineering and has been widely adopted as an IPv6 based SDN solution.

SRv6 creates a network domain with predefined network segments that can be set up within an IPv6 network where segment-based traffic steering is desired.

Timeline

SRv6 support in SAI

SRv6 headend and Endpoint behaviors including uSID

END, END.T, END.X, END.DT4, END.DT6, END.DT46, uN, uA, B6.ENCAP, ….

Nov 2021

SONiC 202111

SRv6 domain ingress|transit|egress nodes

H.Encaps.Red

END.DT4, END.DT6, END.DT46

Anycast SID in END

Traffic steering on SID list

Nov 2021

SONiC 202212

SRv6 uSID

Nov 2021

More SRv6

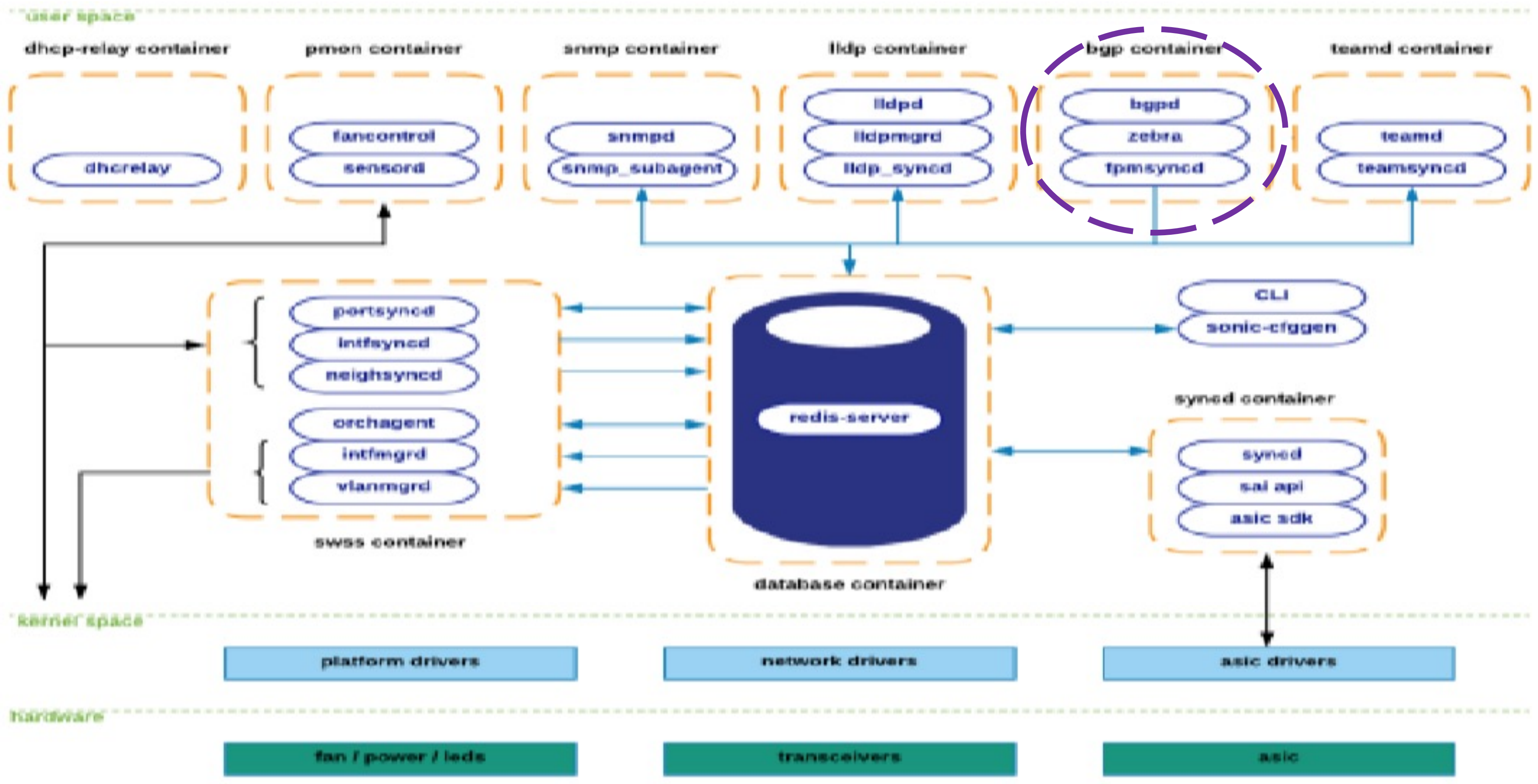
Integration with FRR

sBFD for SRv6

Path Tracing (PT)

2023

SONiC Architecture



SRv6 support in SAI

SRv6 APIs and attributes in-line with the latest RFCs and SAI MPLS/tunnel pipeline model

HeadEnd Behavior

Support both {color, DIP} via ACL & DIP only, via LPM policy lookup

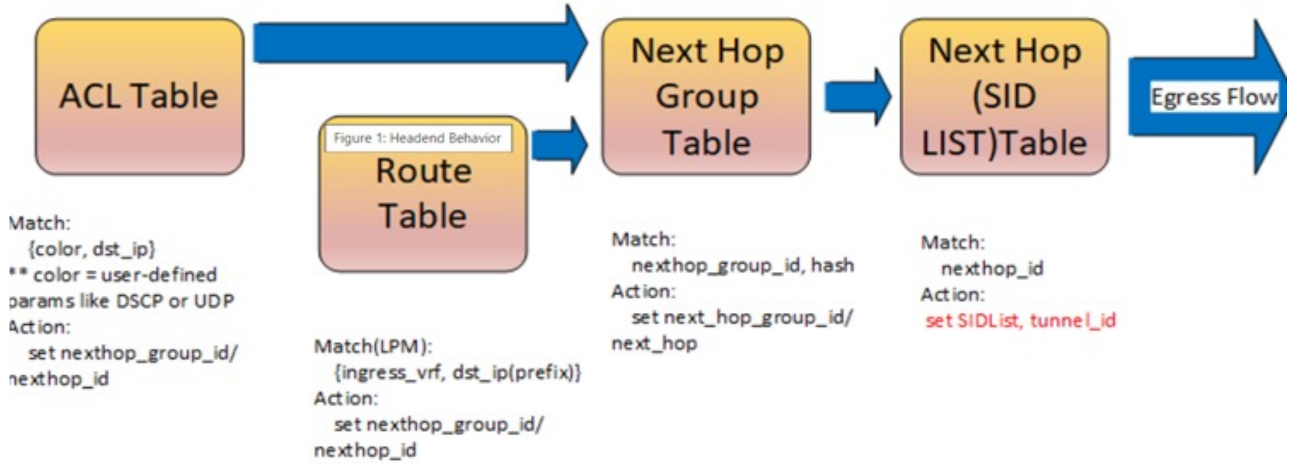


Figure 1: Headend Behavior

ACL Table
Route Table
Next Hop Group Table
Next Hop (SID LIST)Table
Egress Flow
Match: {color, dst_ip} ** color = user-defined params like DSCP or UDP Action: set nexthop_group_id/ nexthop_id
Match(LPM): {ingress_vrf, dst_ip(prefix)} Action: set nexthop_group_id/ nexthop_id
Match: nexthop_group_id, hash Action: set next_hop_group_id/ next_hop
Match: nexthop_id Action: set SIDList, tunnel_id

Endpoint Behavior

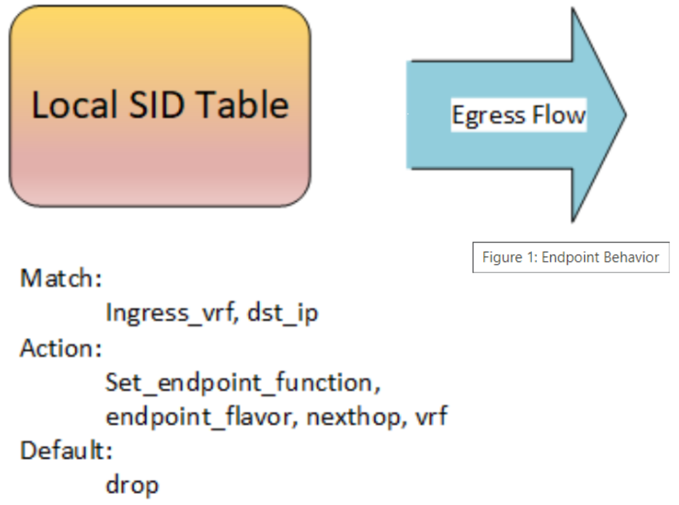


Figure 1: Endpoint Behavior

Local SID Table
Egress Flow
Match: Ingress_vrf, dst_ip
Action: Set_endpoint_function, endpoint_flavor, nexthop, vrf
Default: drop

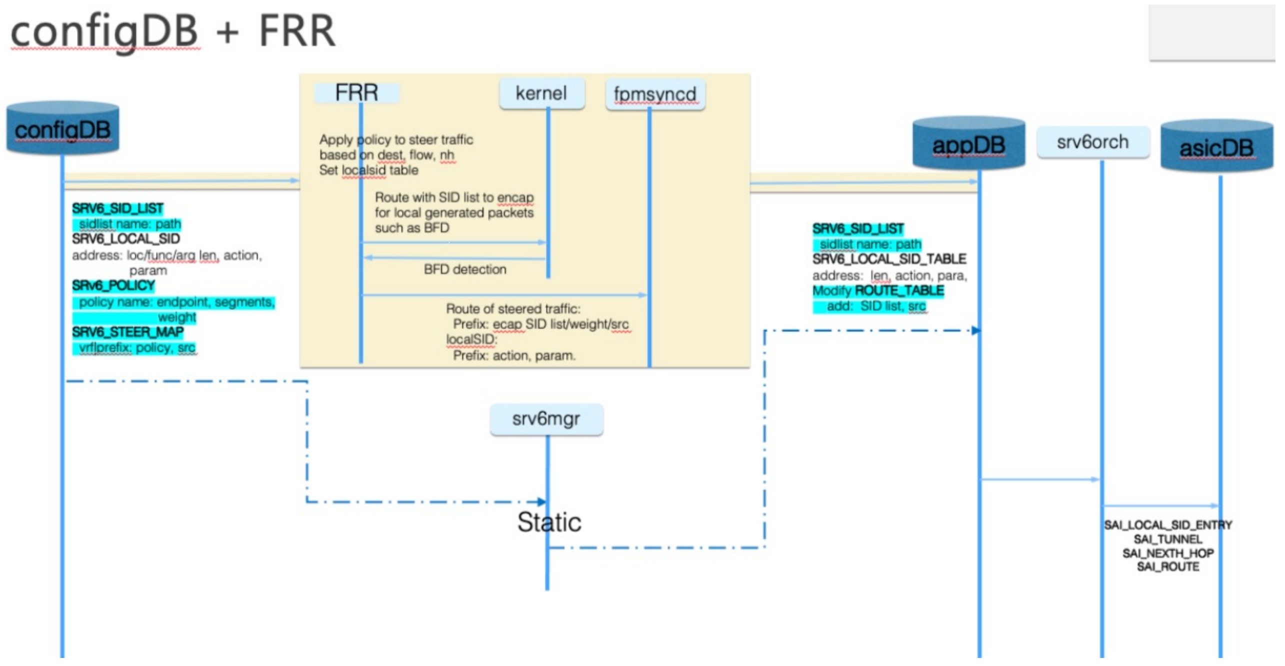
SRv6 implementation in SONiC

Controller based configuration to set SID-List and apply policy for TE in APPL_DB tables.

New Srv6Orch handles tables from APPL_DB
- SRv6_SID_LIST – creates SRv6 SID-list segment
- ROUTE_TABLE – adds route pointing to SID-list segment
- SRV6_MY_SID – create local (my) SID entry for END behavior action

MicroSID (uSID) support is added (New).

Route entry that is SRv6 modified and assign higher priority to do SID and nexthop lookup in SRv6 Orchagent



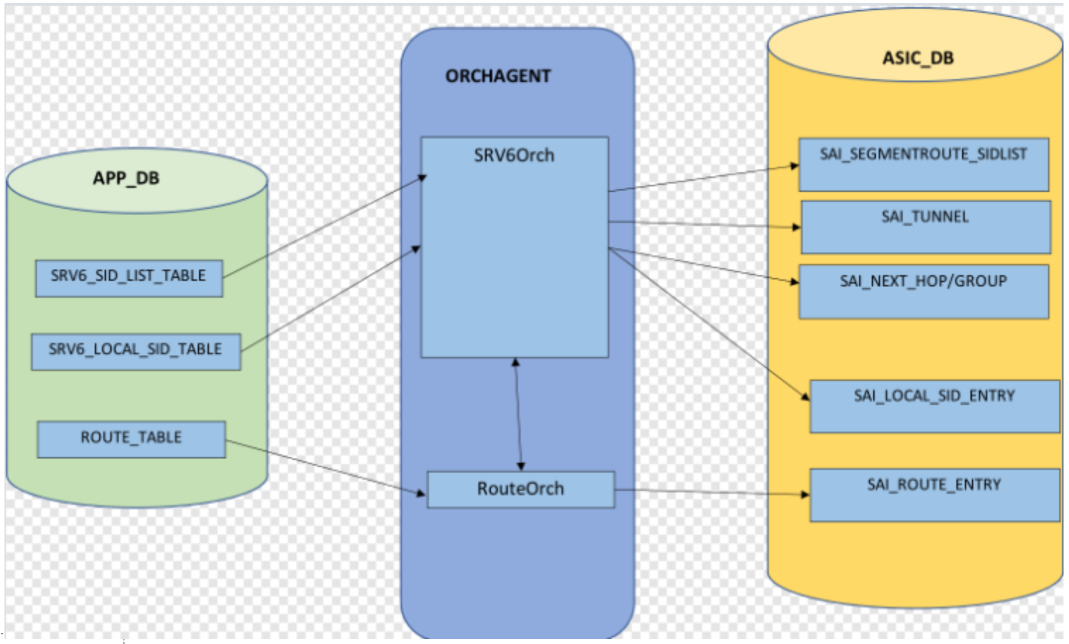
SRv6 Tables in Application DB

New table - SRv6 SID list
key = **SRV6_SID_LIST_TABLE**:segment_name

field = value
path = SID, ; List of SIDs

New table - Local SID to behavior mapping
key = **SRV6_MY_SID_TABLE**:block_len:node_len:func_len:arg_len:ipv6address

field = value
action = behavior ;behaviors defined for local SID
vrf = VRF_TABLE.key ;VRF name for END.DT46, can be empty
adj = address, ; List of adjacencies for END.X, can be empty
segment = SRv6_SID_LIST.key, ; List of segment names for END.B6.ENCAP
source = address, ; List of src addrs for encap for END.B6.ENCAP



key = **ROUTE_TABLE**:VRF_NAME:prefix

nexthop = prefix, ; IP addresses separated ',' (empty indicates no gateway)
intf = ifindex? PORT_TABLE.key ; zero or more separated by ',' (zero indicates no interface)
vni_label = VRF.vni ; zero or more separated by ',' (empty value for non-vxlan next-hops)
router_mac = mac_address ; zero or more remote router MAC address separated by ','
(blackhole = BIT ; Set to 1 if this route is a blackhole (or null0)
segment = SRV6_SID_LIST.key ; New optional field. List of segment names, separated by ','
seg_src = address ; New optional field. Source addrs for sid encap

Generic SAI Extensions

P4 Integrated Network Stack

→ New programmable capabilities

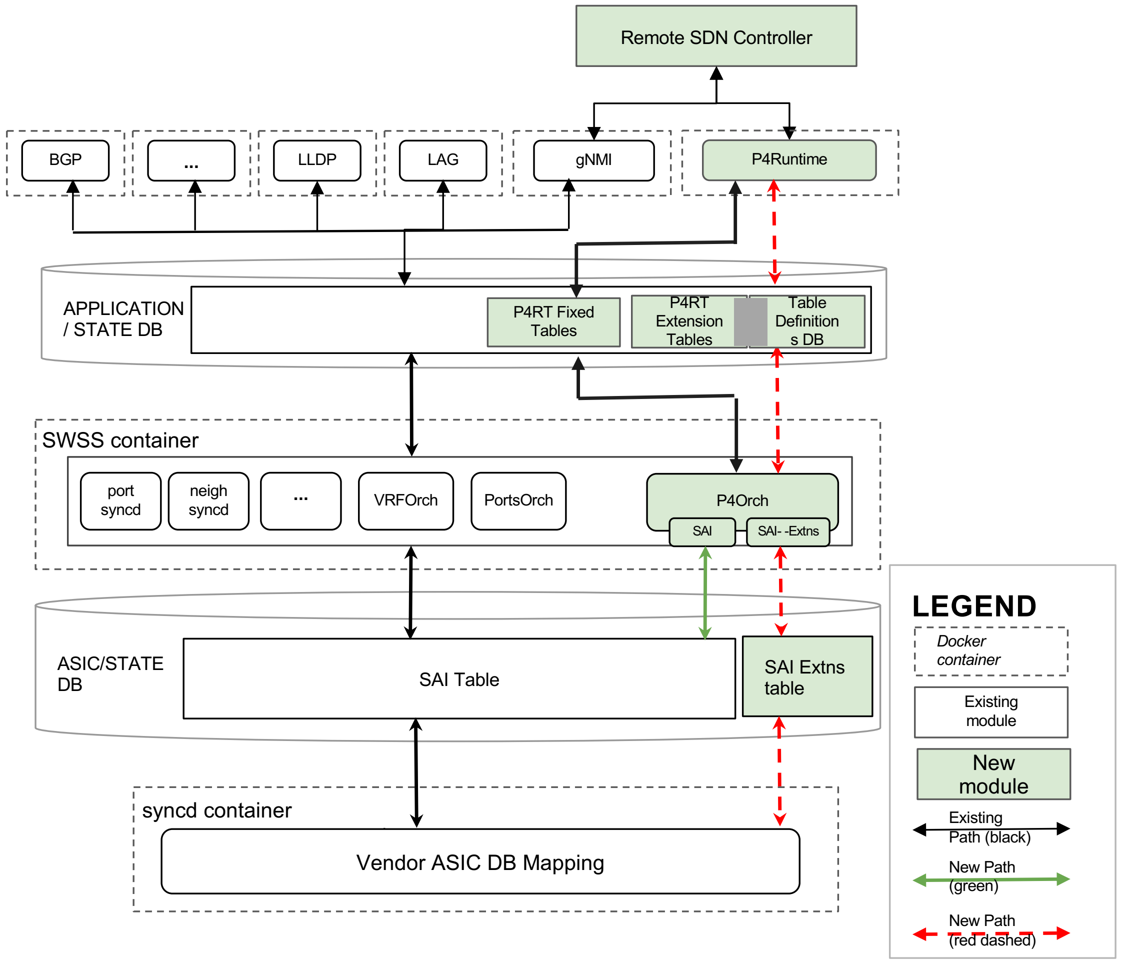
→ Realization of new user scenarios

→ Enable referencing existing SAI objects

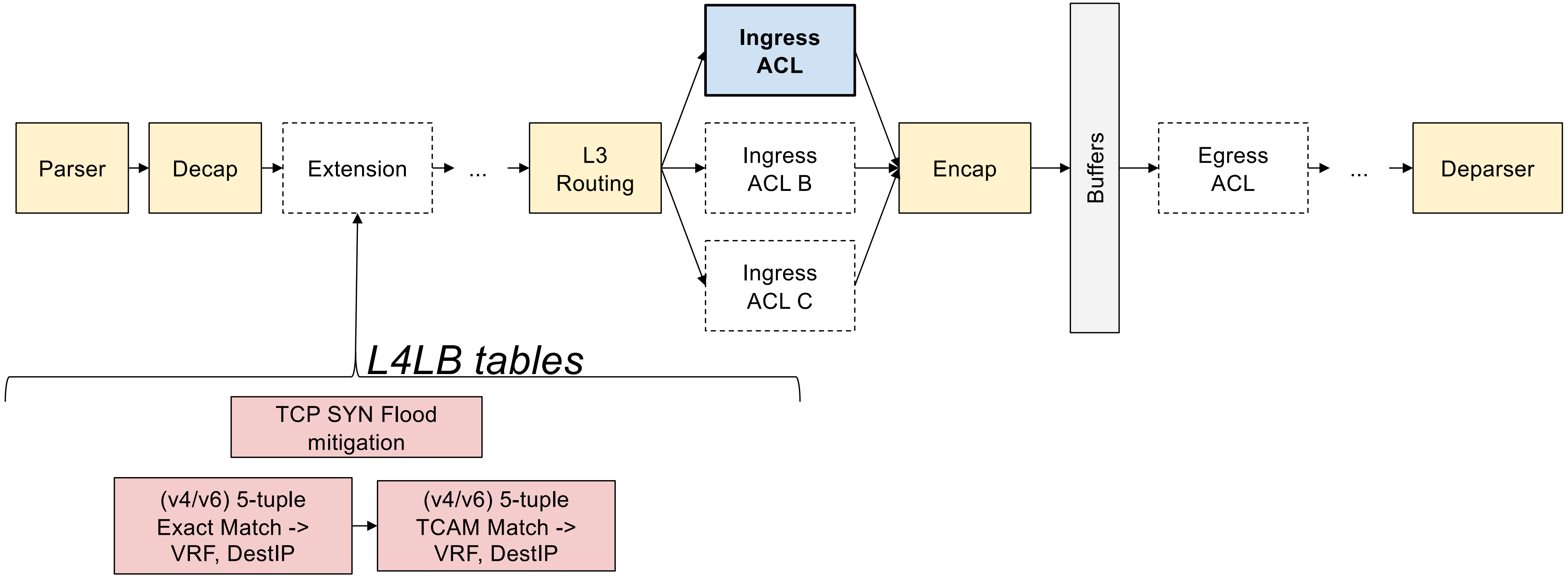
→ Brings agility and differentiation with specialized use cases

P4Orch

- New Extensions manager in P4Orch
- A TableMap database derived from P4 Info to generically support programming of Extension table entries



SRv6 VNF - Load Balancing implementation with GSE



SRv6 in SONiC Ecosystem

Companies contributing to SRv6 in the SONiC community

- Intel
- Cisco
- Alibaba
- Microsoft
- And others in SONiC community

Thank you

Notices & Disclaimers

SRv6 and FRR

Carmine Scarpitta[1]

University of Rome Tor Vergata / CNIT

Linux Netdev 0x16

Who Is This Guy?

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

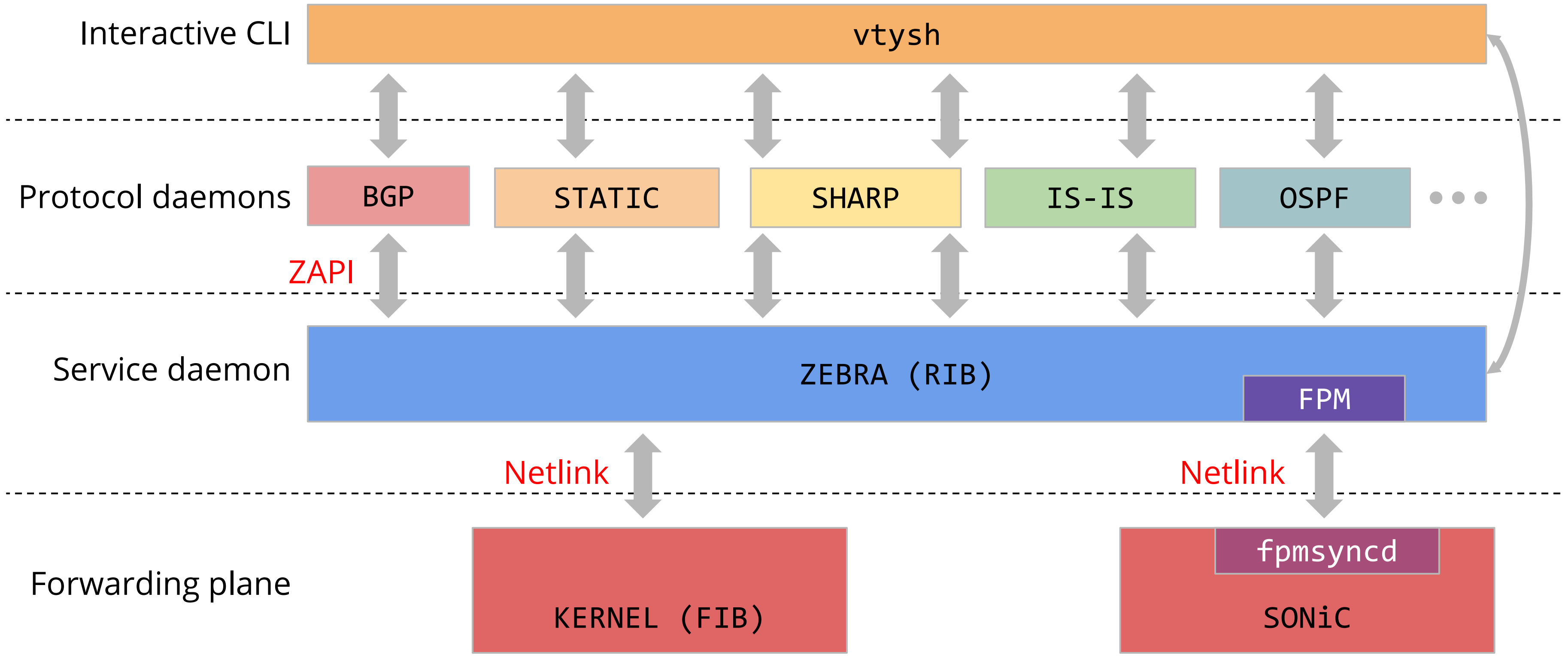
cnit consorzio nazionale interuniversitario per le telecomunicazioni

- I'm **Carmine Scarpitta**
- I'm a **PhD Student** at the University of Rome Tor Vergata
- My research interests include **SRv6**, **SDN**, **SD-WAN**
- I have been working on **FRR** and **SONiC** for 7 months
  - Developing **new features** related to SRv6

Agenda

- Introduction to **FRR Architecture**
- **SRv6 features supported** in mainline of FRR
- **New SRv6 features** that we implemented in FRR
- What’s next?

FRR Architecture

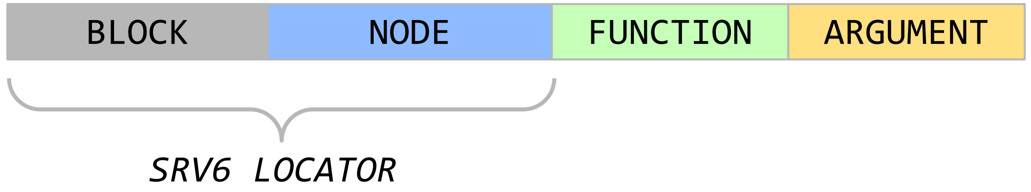


SRv6 Support in FRR and SONiC

- The **SRv6 development** in FRR has started recently
- FRR supports a rich SRv6 ecosystem
  - e.g. , SRv6 Locator, L3VPN services, SRv6 Behaviors, …
- The SRv6 support in FRR is across **several daemons** (Zebra, BGP, SHARP)
- We are continuously adding **new SRv6 features** to FRR!
- We are working on the integration of **our SRv6 implementation** in mainline of FRR
- We are working on the **integration of the SRv6 features in SONiC**

SRv6 Locator (1/2)

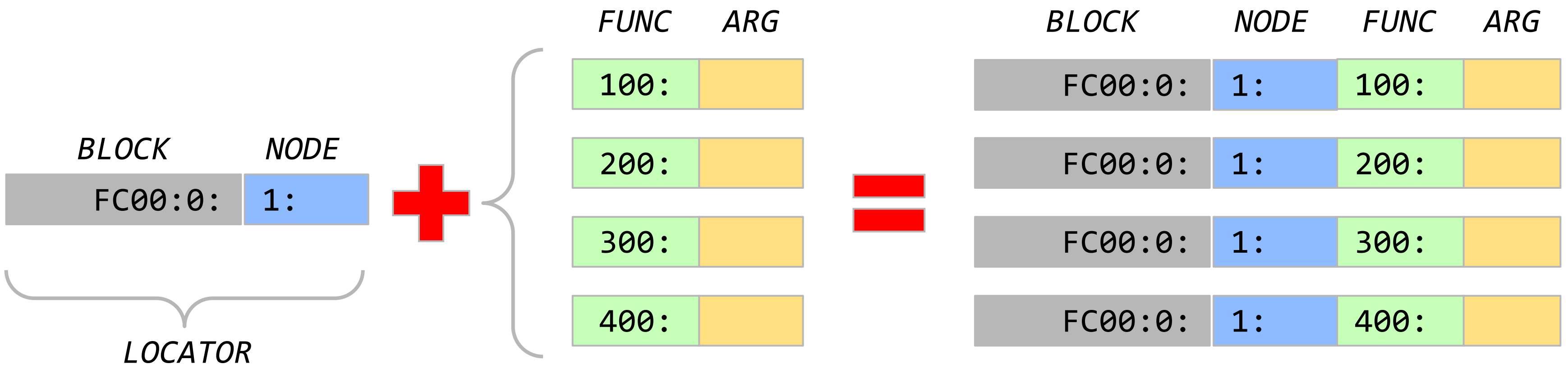
- **SRv6 SID Format** (defined in RFC 8986)



- The RFC gives operators the freedom to choose the **length of SRv6 Locator, Function, etc.**

SRv6 Locator (2/2)

- Typically
  - 1 SRv6 Locator per-node
  - Several SRv6 SIDs allocated from the locator



Support for SRv6 Locator in FRR

- In FRR you can **configure SRv6 Locators** using the vtysh shell

```
router(config)# segment-routing
router(config-sr)# srv6
router(config-srv6)# locators
router(config-srv6-locators)# locator LOC1
router(config-srv6-locator)# prefix fc00:0:1::/48 func-bits 16 block-len 32 node-len 16
router(config-srv6-locator)# behavior usid
```

  - `LOC1` is the name of the locator
  - `func-bits/block-len/node-len` are the lengths of the Function/Block/Node parts of the SIDs allocated from the locator
  - **`behavior usid`** specifies the locator as a uSID locator

- `func-bits` is already supported by the mainline implementation of FRR

- We have an open PR to integrate **`block-len`** and **`node-len`** in mainline FRR
  - https://github.com/FRRouting/frr/pull/11673

- **`behavior usid`** is implemented on our fork of FRR
  - https://github.com/cscarpitta/frr (branch srv6-usid) - we will open a PR to integrate it on mainline

BGP Overlay Services Based on SRv6

- We want to support **L3VPN Services** in SONiC using
  - BGP as control-plane
  - SRv6 as data-plane
- RFC 9252 extends BGP to **support SRv6 services** (e.g., L3VPN and EVPN)
- **BGP** is used to **advertise the reachability of prefixes** of a particular VPN from an egress PE to the ingress PE nodes
  - The egress PE signals an SRv6 Service SID
  - The ingress PE encapsulates the payload in an outer IPv6 header where the destination address is the SRv6 Service SID provided by the egress PE
- RFC 9252 introduces **new TLVs** and **Sub-TLVs** (see next slide)
  - Attached to the BGP Update messages to carry SRv6 SID information

BGP Overlay Services Based on SRv6 (TLVs)

- The SID information is enclosed in a **SRv6 L3 Service TLV**
- **SRv6 SID Information Sub-TLV** defines the properties of the SID
  - SRv6 SID Value (e.g., FC00:0:1:100::)
  - SRv6 Endpoint Behavior codepoint (e.g., 63 for uDT4 or 62 for uDT6)
- **SRv6 SID Structure Sub-Sub-TLV** defines the SID structure
  - Block length
  - Node length
  - Function length
  - Argument length

BGP Overlay Services Based on SRv6 in FRR

- To configure a L3VPN Service in FRR, you need to configure
  - The egress PE to **advertise** the L3VPN prefixes using an SRv6 SID
  - The ingress PE to **import** the advertised prefixes in the appropriate VRF
- The SRv6 Architecture allows to support both IPv4 and IPv6 address families using a single SID
  - uDT46 behavior
- We implemented the support for uDT46 behavior
- We have an open PR to integrate the uDT46 behavior in the mainline FRR
  - https://github.com/FRRouting/frr/pull/11673

Source IPv6 Address for Encapsulated Packets

- We want a mechanism to set the source IPv6 address for the SRv6 packets
- In Linux, you can **use `iproute2`** to **configure a Source IPv6 address**
  - `ip sr tunsrc set FC00:0:1::1`
  - This approach is specific for Linux!
- In FRR you can configure the src IPv6 address both in Linux and SONiC with vtysh

```
router(config)# segment-routing
router(config-sr)# srv6
router(config-srv6)# encapsulation
router(config-srv6-encap)# source-address fc00:0:1::1
```

- In the mainline distribution of FRR this command is not yet supported
- But we have an implementation on our fork of FRR
  - https://github.com/cscarpitta/frr (branch srv6-usid)
- We will integrate this functionality in the mainline of FRR

Support for SRv6 Behaviors in STATIC

TOR VERGATA UNIVERSITÀ DEGLI STUDI DI ROMA

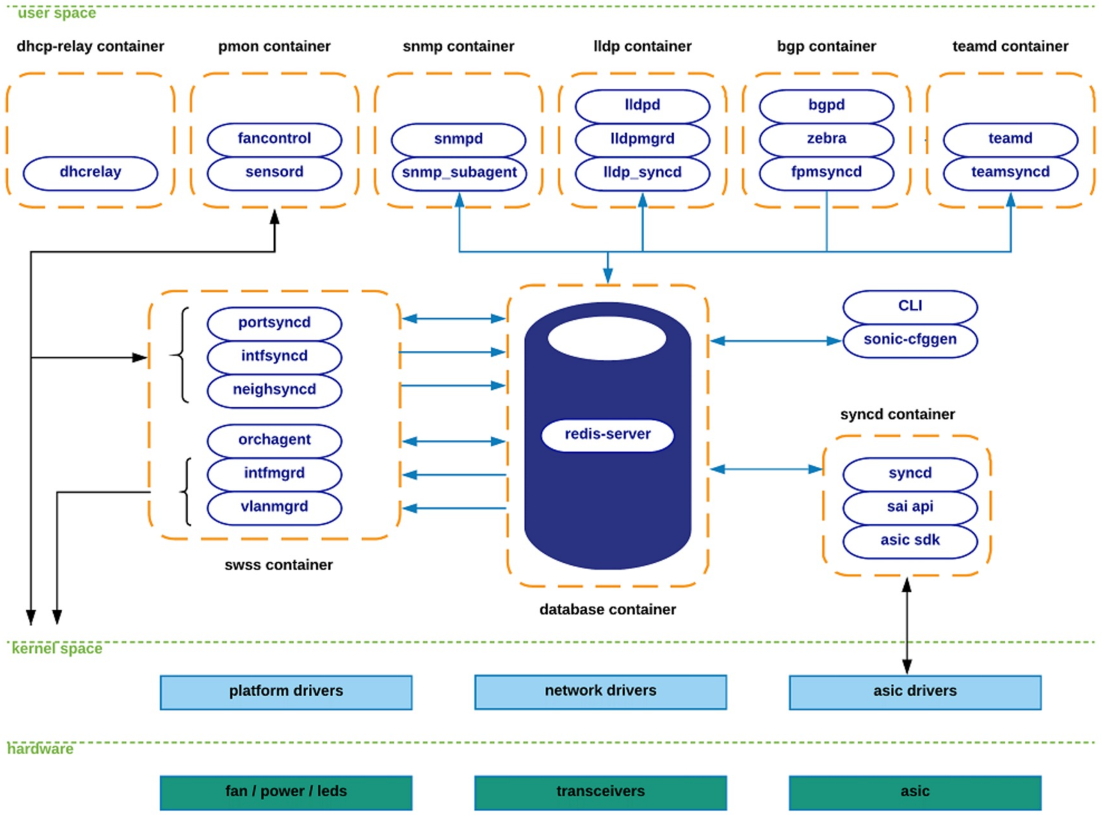
cnit consorzio nazionale interuniversitario per le telecomunicazioni

- **STATIC** is a daemon that handles the installation/deletion of static routes
- In FRR you can configure the SRv6 Endpoint Behaviors in FRR and SONiC with vtysh

```
router(config)# segment-routing
router(config-sr)# srv6
router(config-srv6)# explicit-sids
router(config-srv6-sids)# sid fcbb:bb00:1:f001 behavior end-dt6-usid
router(config-srv6-sid)# sharing-attributes
router(config-srv6-sid-attr)# vrf-name VRFA
```

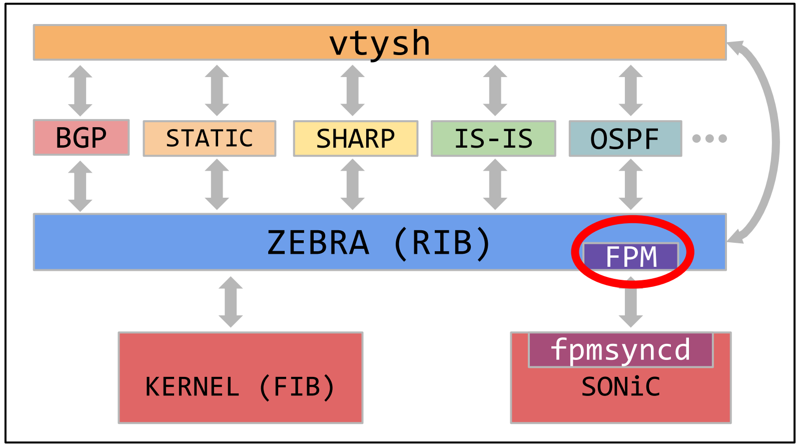
- This feature is not in the mainline implementation of FRR
- We have an open source implementation on our fork of FRR
  - https://github.com/cscarpitta/frr (branch srv6-usid) - we will open a PR to integrate it on mainline
  - currently only a subset of behaviors including **uDT4**, **uDT6**, and **uDT46**
  - WIP: we are implementing other behaviors
- We will integrate this feature in the mainline FRR!

SONiC Architecture



Exporting SRv6 Behaviors to SONiC (1/3)

- Routing daemons compute their best routes and send these route to zebra through the ZAPI
- zebra computes the best routes across the routing daemons and install these routes both in the kernel and in SONiC.
- fpmsyncd in SONiC listens on the FPM port (2620)
- FPM periodically attempts to connect to fpmsyncd
- Once the connection is up, zebra sends a complete copy of the forwarding table to the fpmsyncd (routes encoded in Netlink format)



Exporting SRv6 Behaviors to SONiC (2/3)

- FPM should be able to **encode** the SRv6 Behaviors in Netlink format and **push** them to fpmsyncd
- The mainline implementation of FRR does not support encoding SRv6 Behaviors
- We have an open source implementation of this feature on our fork
  - https://github.com/cscarpitta/frr (branch srv6-usid)
- We will integrate our implementation in the mainline FRR

Exporting SRv6 Behaviors to SONiC (3/3)

- **fpmSyncd** in SONiC waits for notification about route additions and deletions
  - … also events related to SRv6 - e.g., installing/deleting SRv6 Behaviors!
- fpmsyncd parses the received update messages and delivers each message to the appropriate **handler**
  - SRv6-related messages are delivered to SRv6 handlers
- fpmsyncd publishes route changes and SRv6 behaviors changes to **Redis** in real time
- Redis propagates the changes to syncd
- Syncd updates the FIB
- The mainline SONiC does not support the SRv6 handlers to decode SRv6-related messages
- We have an implementation of this feature on our fork of SONiC
  - https://github.com/cscarpitta/sonic-swss (branch srv6-usid)
- We will integrate our implementation in the mainline of SONiC

Conclusions (1/2)

- The mainline distribution of FRR supports the **several SRv6 features**
  - Configuring SRv6 Locators
  - Instantiating a subset of SRv6 behaviors
- We implemented many additional features (available on our fork of FRR)
  - Configuring micro-segment SRv6 Locators
  - Setting a Source IPv6 Address for SRv6 packets
  - Exporting L3VPN Services using BGP with a single SID
  - Supporting more SRv6 behaviors in SHARP
  - Introduced the support for SRv6 to STATIC
  - Extended FPM to export SRv6 routes/behaviors to other forwarding planes (e.g., SONiC)
- **We started integrating the new features in the FRR Mainline…**
  - **… and we continue developing new features!**

Conclusions (2/2)

- What's Next?
  - Continue **integrating the features** we developed in the **FRR Mainline**
  - Continue the integration of SRv6 features in SONiC
  - Support **more SRv6 features in FRR**
    - Support for more SRv6 Behaviors
    - Support SRv6 traffic steering

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

consorzio nazionale interuniversitario per le telecomunicazioni

Thank you for your attention!

carmine.scarpitta@uniroma2.it