SRv6でサービスチェイニングをやってみた

POC of SRv6 service chaining

次世代技術本部　桑田 斉

hitoshi.kuwata.gt@apresiasystems.co.jp

JANOG43　2019.1.23.

Motivation

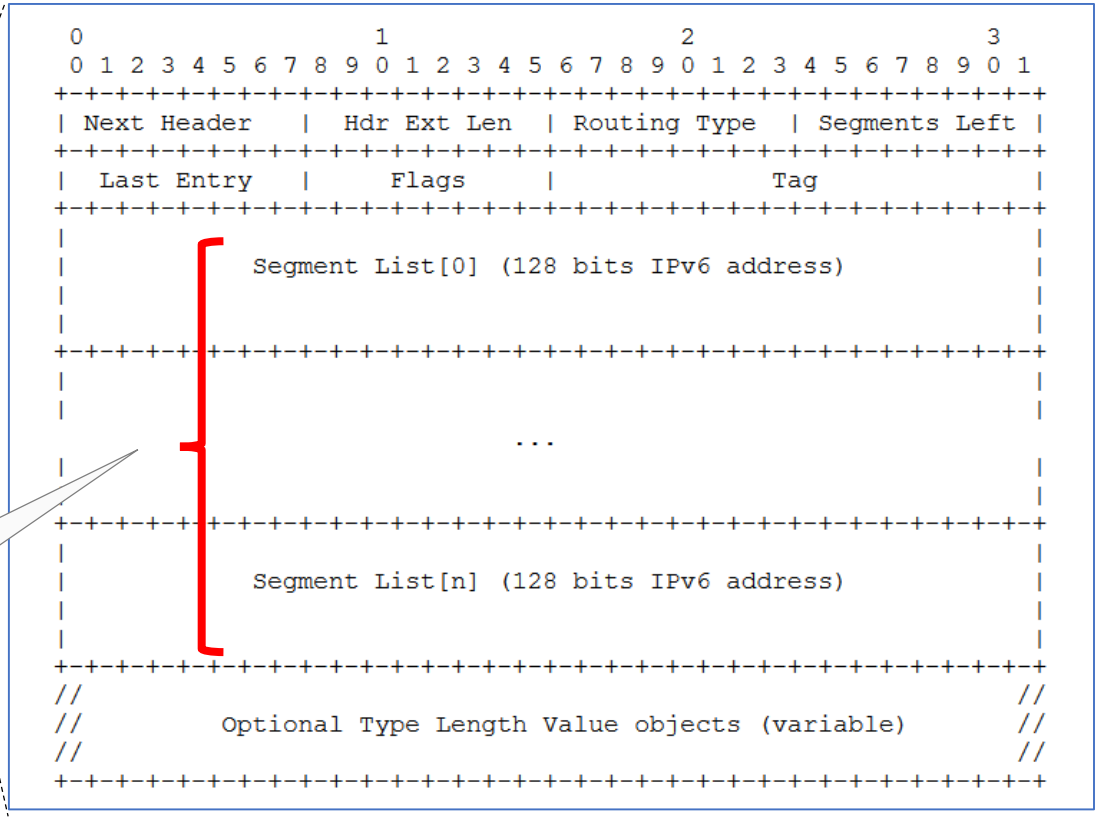
- Can we run SRv6 function already?
  - SRv6 data plane functions have been implemented in Linux.
- How can we run it?
- What can we realize with SRv6?
  - such as service chaining?

------> Let's try!

SRv6 Header format

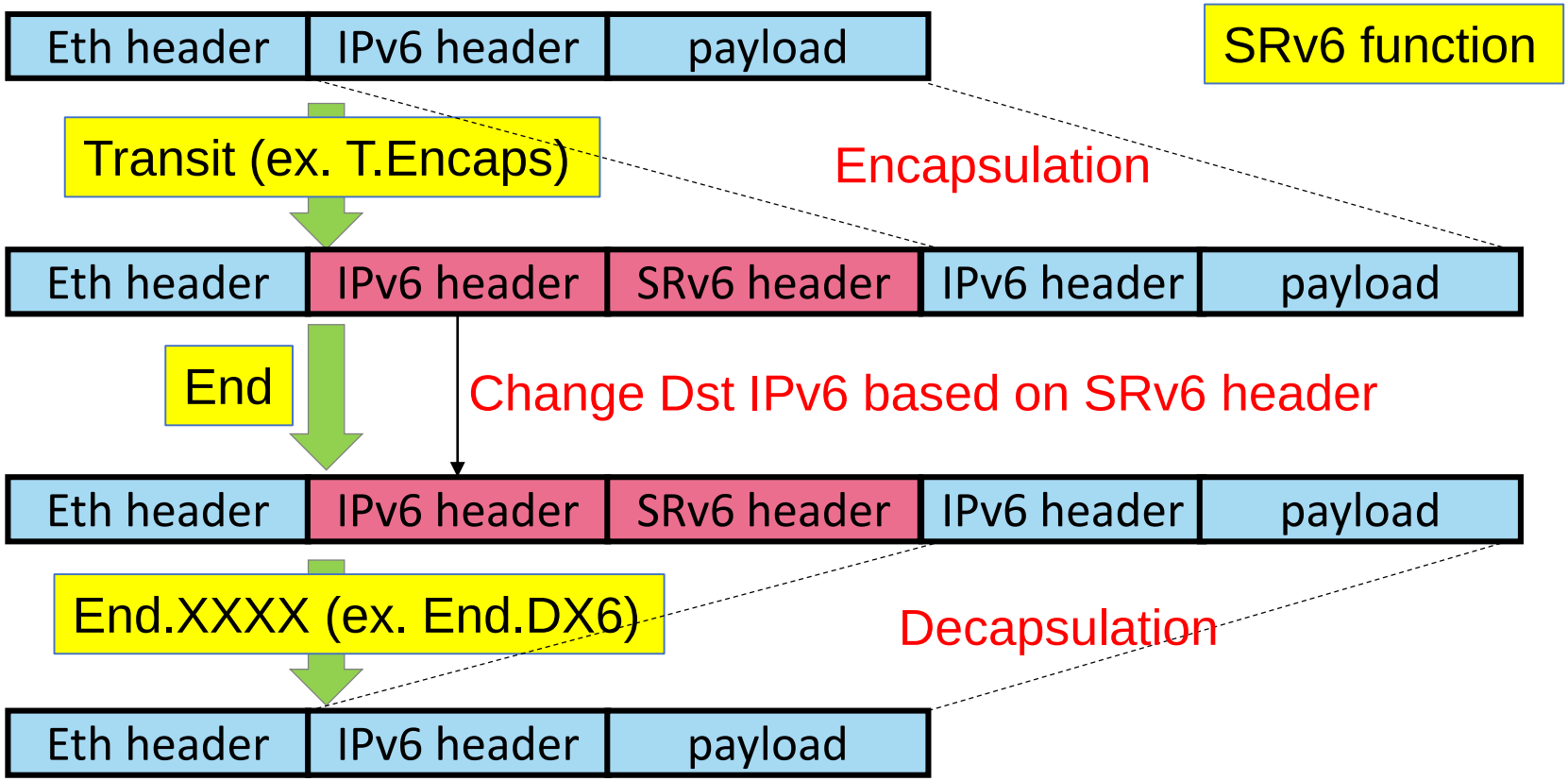
Ether Header
IPv6 Header
IPv6 Optional Header
Payload

IPv6 address format

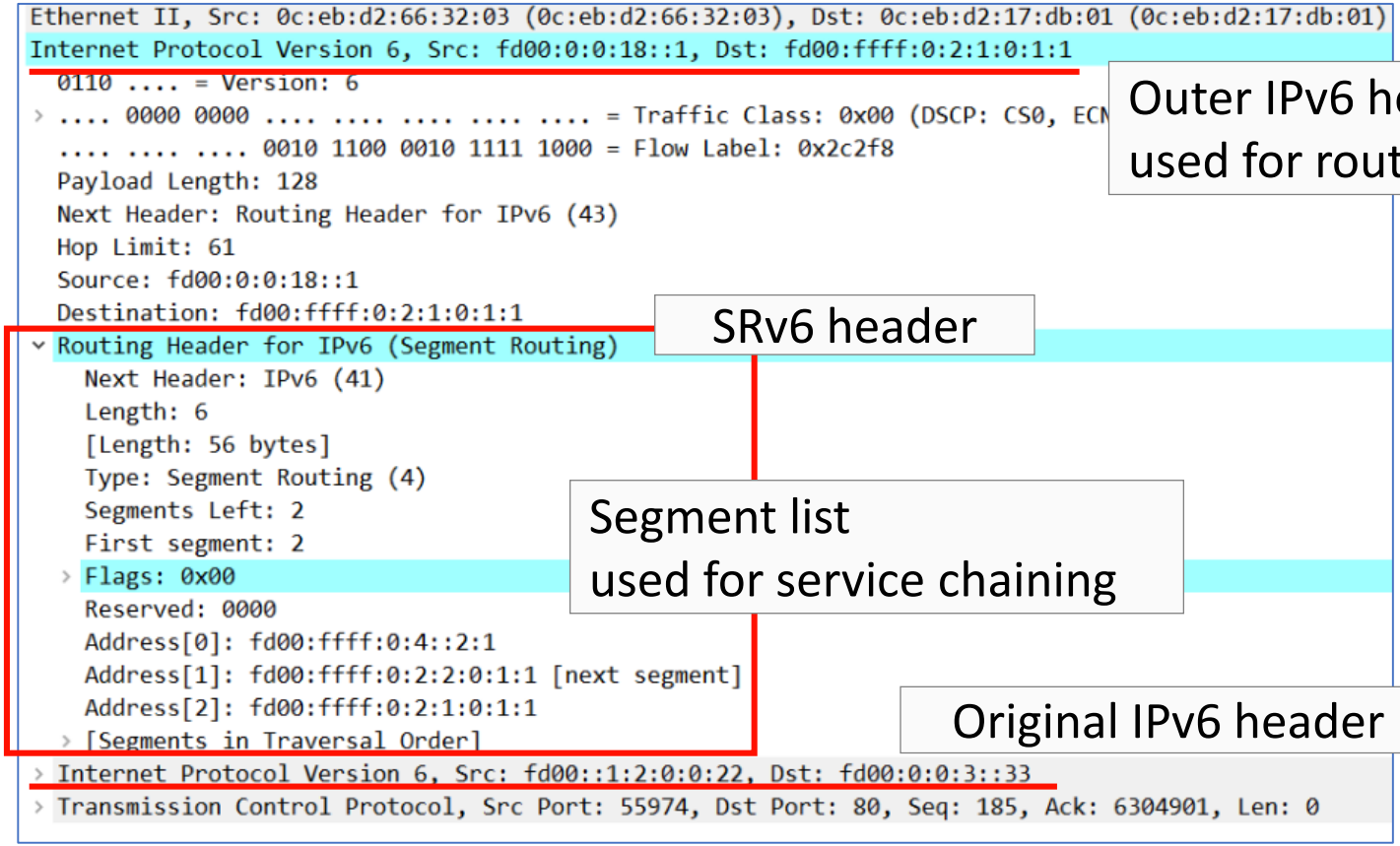


https://tools.ietf.org/html/draft-ietf-6man-segment-routing-header-14

How SRv6 data plane works

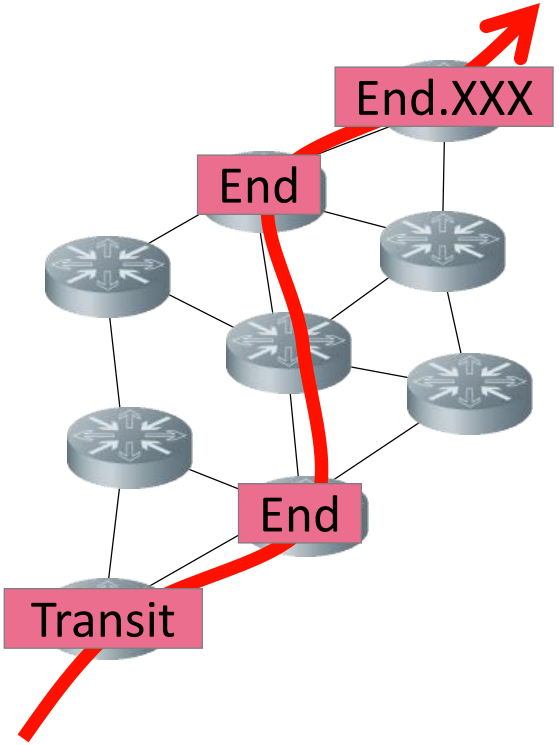


Sample capture of SRv6 using T.Encaps

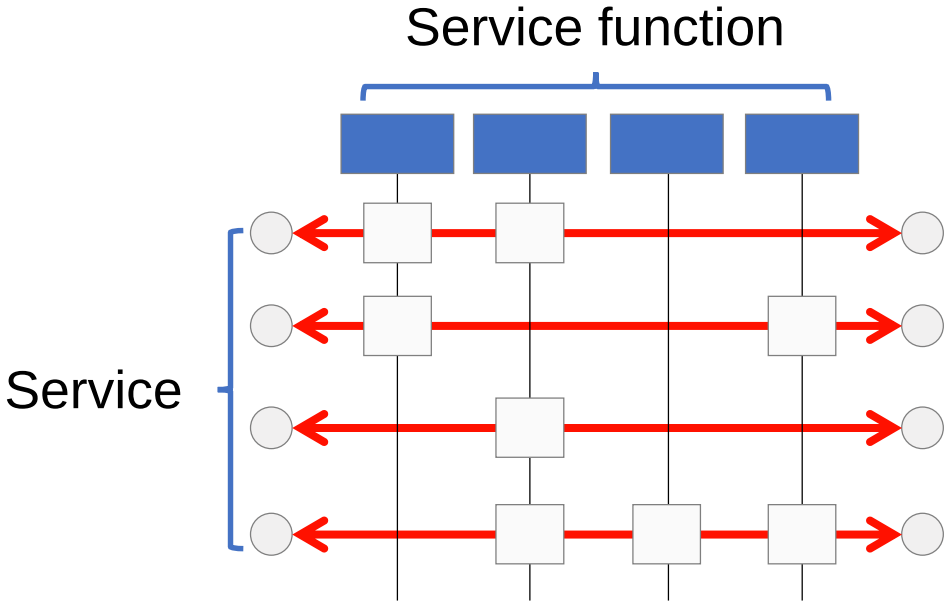


Use case of SRv6

Traffic engineering

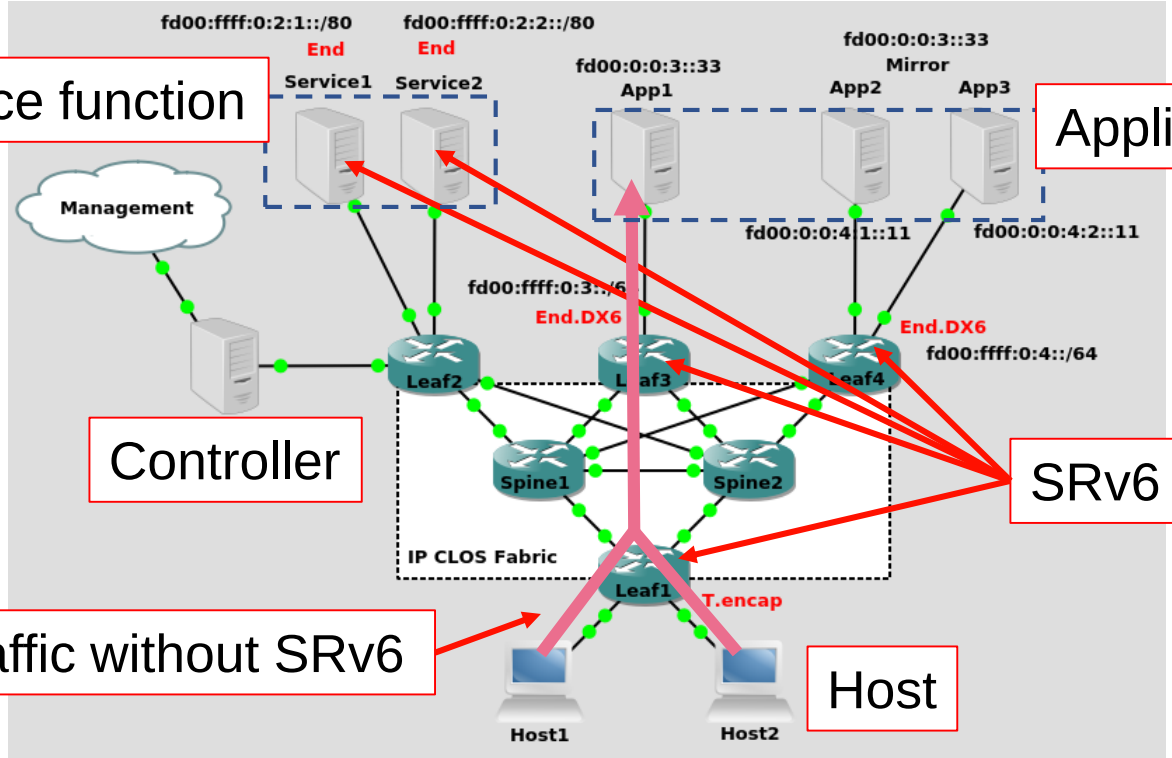


Service chaining



Today's topic

Demo environment on GNS3



Service function

Application

Controller

SRv6 works here

User traffic without SRv6

Host

- MP-BGP-based IP CLOS Fabric
- Service chaining with SRv6

} FRRouting + Linux kernel

FRRouting

- Control plane OSS forked from Quagga
  - https://frrouting.org/
  - https://github.com/frrouting/frr
    - Stable/6.0 on Ubuntu18.04 + Kernel 4.15 is used for this demo.
- One project of Linux Foundation Networking

https://www.linuxfoundation.jp/projects/networking/

Current development status of Linux kernel

- Linux kernel 4.14 or later can run several SRv6 functions.

End function

Name	Description	Release
End	Endpoint function	4.10 (February 2017), srext
End.X	Endpoint function with Layer-3 cross-connect	4.10 (February 2017), srext
End.T	Endpoint function with specific IPv6 table lookup	4.14 (November 2017)
End.DX2	Endpoint with decapsulation and Layer-2 cross-connect	4.14 (November 2017), srext
End.DX6	Endpoint with decapsulation and IPv6 cross-connect	4.14 (November 2017), srext
End.DX4	Endpoint with decapsulation and IPv4 cross-connect	4.14 (November 2017), srext
End.DT6	Endpoint with decapsulation and IPv6 table lookup	4.14 (November 2017)
End.DT4	Endpoint with decapsulation and IPv4 table lookup	In development
End.B6	Endpoint bound to an SRv6 policy	4.14 (November 2017)
End.B6.Encaps	Endpoint bound to an SRv6 encapsulation Policy	4.14 (November 2017)
End.BM	Endpoint bound to an SR-MPLS Policy	In development
End.S	Endpoint in search of a target in table T	In development
End.AD	Endpoint to SR-unaware APP via dynamic proxy	srext
End.AM	Endpoint to SR-unaware APP via masquerading	srext

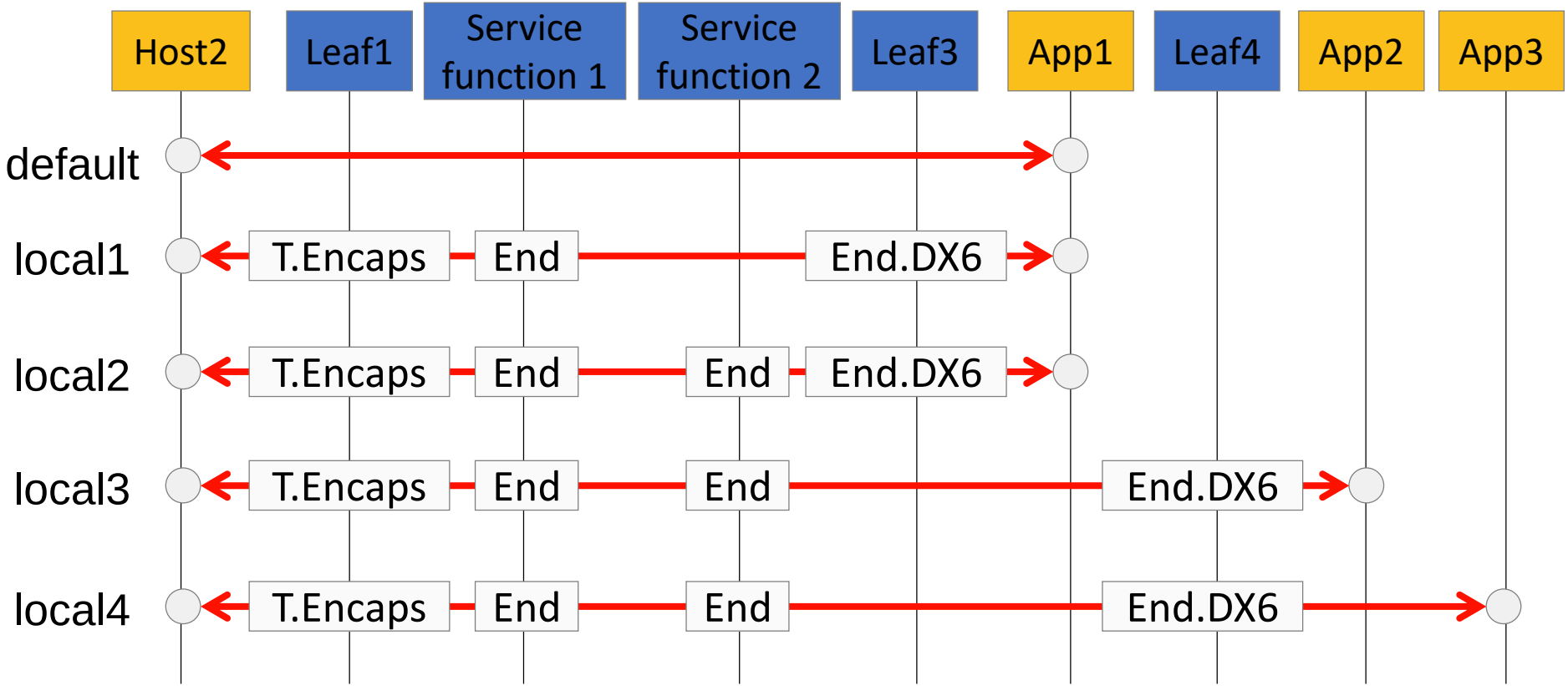
Transit function

Name	Description	Release
T.Insert	Transit behavior with insertion of an SRv6 Policy	4.10 (February 2017)
T.Encaps	Transit behavior with encapsulation in an SRv6 policy	4.10 (February 2017)
T.Encaps.L2	T.Encaps behavior of the received L2 frame	4.14 (November 2017)

Used for this demo

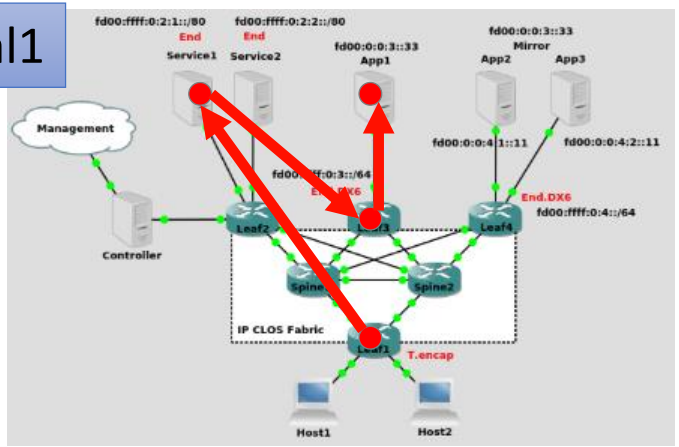
http://www.segment-routing.net/open-software/linux/

Configuration of SRv6 service chaining

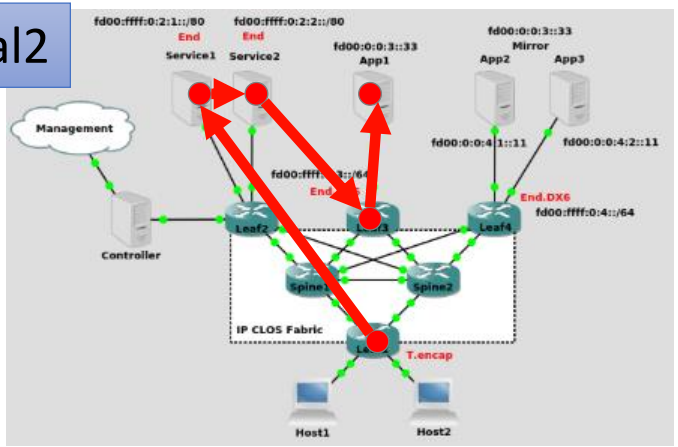


Pattern of service chaining on this demo

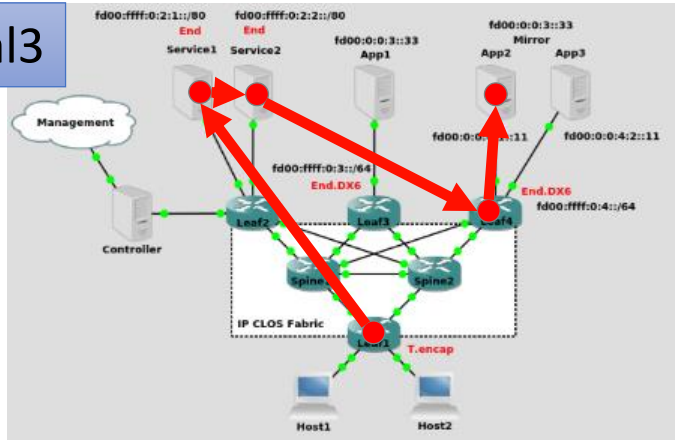
local1



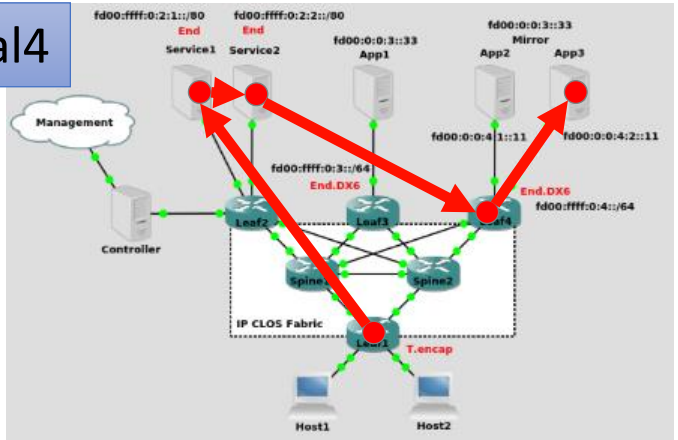
local2



local3

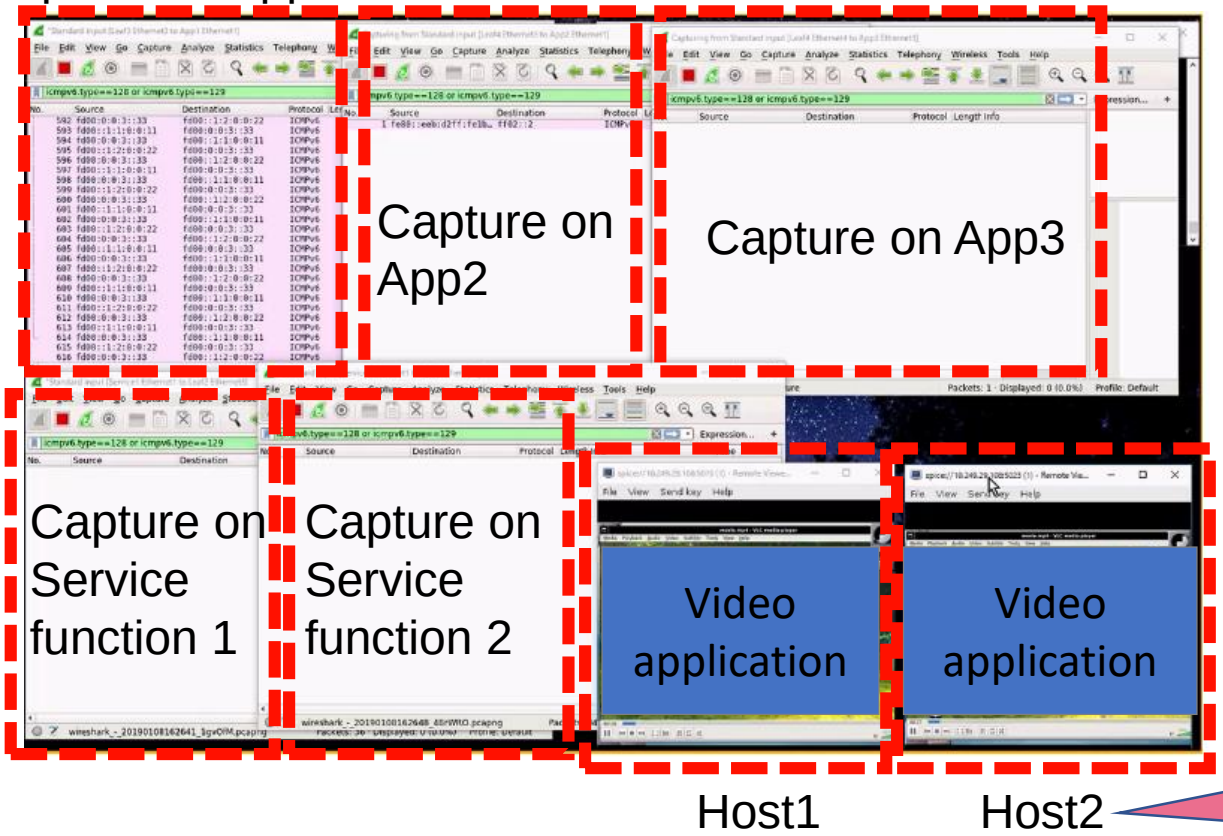


local4



Demo screenshot

Capture on App1



Traffic steering with SRv6 Service Chaining

Pre-configurations of End, End.DX6

Location	configuration
Service function 1	sudo ip -6 route add fd00:ffff:0:2:1::1:1 encap seg6local action End dev ens4
Service function 2	sudo ip -6 route add fd00:ffff:0:2:2::1:1 encap seg6local action End dev ens4
Leaf3	sudo ip -6 route add fd00:ffff:0:3::2:1 encap seg6local action End.DX6 nh6 :: dev ens6
Leaf4	sudo ip -6 route add fd00:ffff:0:4::2:1 encap seg6local action End.DX6 nh6 fd00:0:0:4:1::11 dev ens6 fd00:0:0:4:1::11 == App2
Leaf4	sudo ip -6 route add fd00:ffff:0:4::2:2 encap seg6local action End.DX6 nh6 fd00:0:0:4:2::11 dev ens6 fd00:0:0:4:2::11 == App3

How App2 and App3 can acts as App1

- App2 and App3 have the same address in lo interface as APP1.

```
Netplan configuration for App2

network:
  version: 2
  renderer: networkd
  ethernets:
    ens4:
      dhcp4: false
      addresses: ['fd00:0:0:4:1::11/80']
      gateway6: 'fd00:0:0:4:1::1'
    lo:
      dhcp4: false
      addresses: ['fd00:0:0:3::33/64']
```

Next hop address for End.DX6

App1's IPv6 address

Pre-configurations of T.Encaps

location	Segment list	configuration
Leaf1	Service function 1 -> App1	sudo ip -6 route add fd00:0:0:3::/64 encap seg6 mode encap segs fd00:ffff:0:2:1::1:1,fd00:ffff:0:3::2:1 dev ens4 table local1
	Service function 1 -> Service function 2 -> App1	sudo ip -6 route add fd00:0:0:3::/64 encap seg6 mode encap segs fd00:ffff:0:2:1::1:1,fd00:ffff:0:2:2::1:1,fd00:ffff:0:3::2:1 dev ens4 table local2
	Service function 1 -> Service function 2 -> App2	sudo ip -6 route add fd00:0:0:3::/64 encap seg6 mode encap segs fd00:ffff:0:2:1::1:1,fd00:ffff:0:2:2::1:1,fd00:ffff:0:4::2:1 dev ens4 table local3
	Service function 1 -> Service function 2 -> App3	sudo ip -6 route add fd00:0:0:3::/64 encap seg6 mode encap segs fd00:ffff:0:2:1::1:1,fd00:ffff:0:2:2::1:1,fd00:ffff:0:4::2:2 dev ens4 table local4

How to select service chaining

- Only to do is running them step-by-step at Leaf 1
  - sudo ip -6 rule add from fd00:0:0:1:2::22 table local1
  - sudo ip -6 rule add from fd00:0:0:1:2::22 table local2
  - sudo ip -6 rule add from fd00:0:0:1:2::22 table local3
  - sudo ip -6 rule add from fd00:0:0:1:2::22 table local4
    - fd00:0:0:1:2::22 == Host2

- There is no need to change routing configuration of underlay network and host environment.

Next step?

- Hardware-based SRv6 data plane?
  - Such as P4 programmable switch.

Wedge100BF-32X

Barefoot/Tofino
QSFP28 x 32 port

Wedge100BF-65X

Barefoot/Tofino
QSFP28 x 65 port

Thank you!