cisco

RIPE 76
Marseille, France
14 - 18 May 2018

SRv6 Network Programming

Network as a computer and deployment use-cases

Pablo Camarillo - Software Engineer
pcamaril@cisco.com

RIPE 76

@SegmentRouting

Agenda

1. SRv6 101
2. SRv6 LocalSIDs functions
3. Deployment use-cases
4. VPN Overlay
5. Service Programming
6. Spray
7. SD-WAN
8. 5G and network slicing

Industry at large backs up SR

Strong customer adoption
WEB, SP, DC, Metro, Enterprise

De-facto SDN Architecture

Standardization
IETF

Multi-vendor Consensus

Open Source
Linux, VPP

Bell SoftBank vodafone orange Alibaba.com Google Walmart

Segment Routing

- Source Routing
  - the topological and service (NFV) path is encoded in packet header
- Scalability
  - the network fabric does not hold any per-flow state for TE or NFV
- Simplicity
  - automation: TILFA sub-50msec FRR
  - protocol elimination: LDP, RSVP-TE, NSH…
- End-to-End
  - DC, Metro, WAN

Two dataplane instantiations

Segment Routing

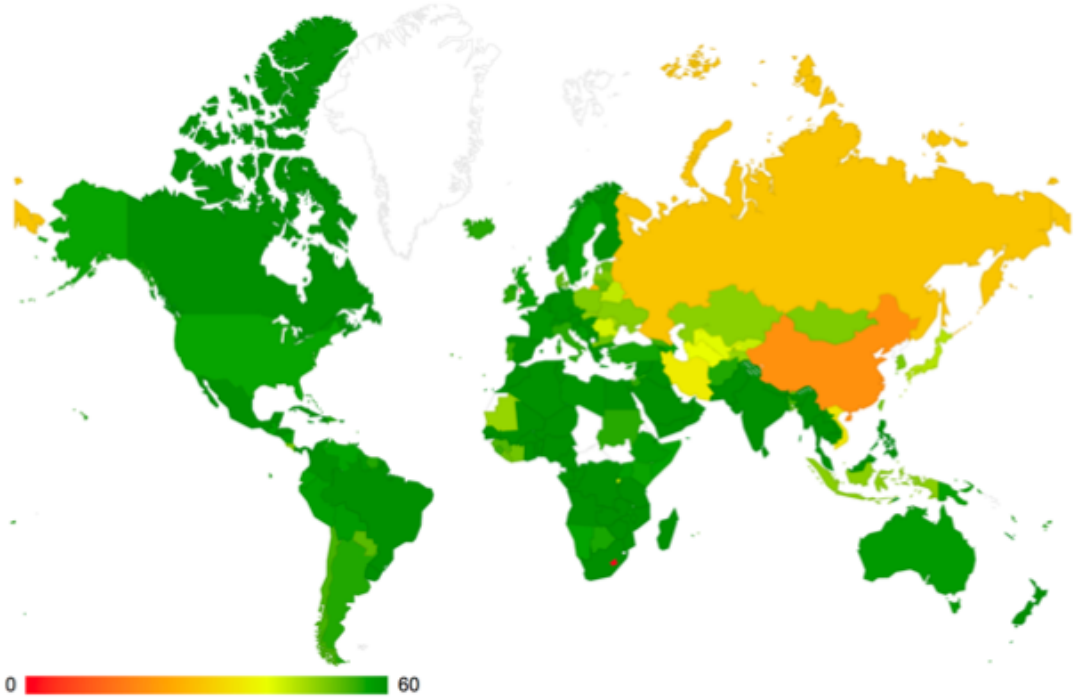
MPLS

- leverage the mature MPLS HW with only SW upgrade
- 1 segment = 1 label
- a segment list = a label stack

IPv6

- leverages RFC8200 provision for source routing extension header
- 1 segment = 1 address
- a segment list = an address list in the SRH

IPv6 adoption is a reality



% Web pages available over IPv6

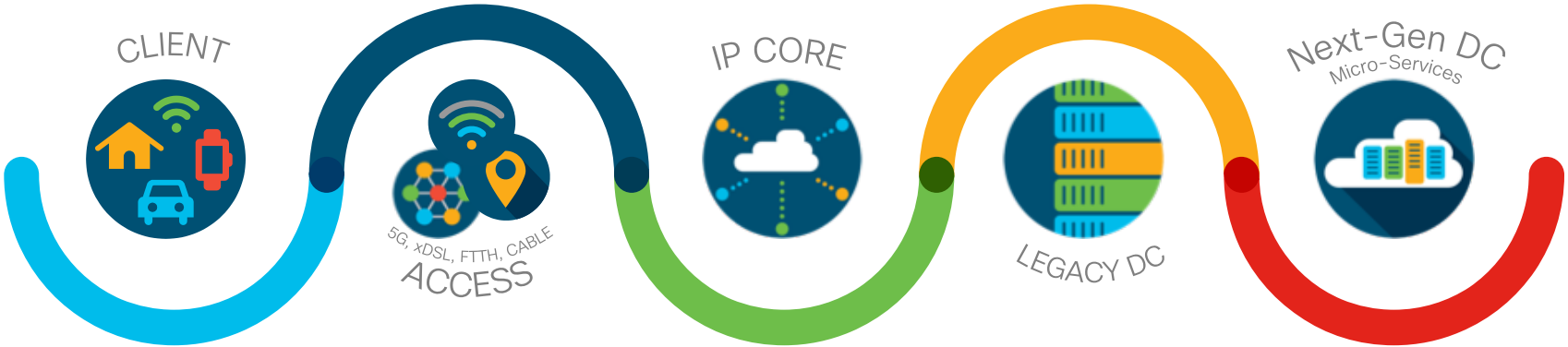
Global IPv6 traffic **grew 241%** in 2016

Globally IPv6 traffic **will grow 16-fold** from 2016 to 2021

IPv6 **will be 37%** of total Internet traffic in 2021

Sources: 6lab.cisco.com – Web content
Cisco VNI Global IP Traffic Forecast, 2016-2021

IPv6 provides reachability



SRv6 – Segment Routing & IPv6

SR for anything else

IPv6 for reachability

- Simplicity
  - Protocol elimination
- SLA
  - FRR and TE
- Overlay
- NFV
- SDN
  - SR is de-facto SDN architecture
- 5G

SRv6 for underlay

SRv6 for Underlay

IPv6 for reachability

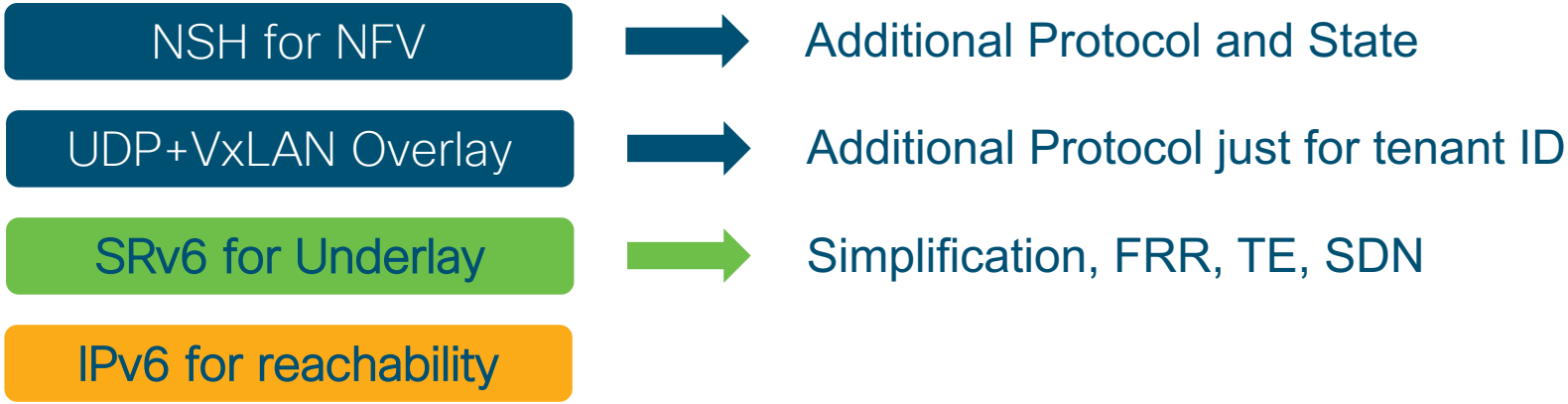
Simplification through protocol reduction

SLA through automated FRR and TE

Horrendous states scaling in k*N^2

De-facto SDN architecture

SRv6 for underlay and overlay



Multiplicity of protocols and states hinder network economics

SR for anything:
Network as a Computer

Network instruction

- 128-bit SRv6 SID
  - Locator: routed to the node performing the function
  - Function: any possible function
    either local to NPU or app in VM/Container
  - Flexible bit-length selection

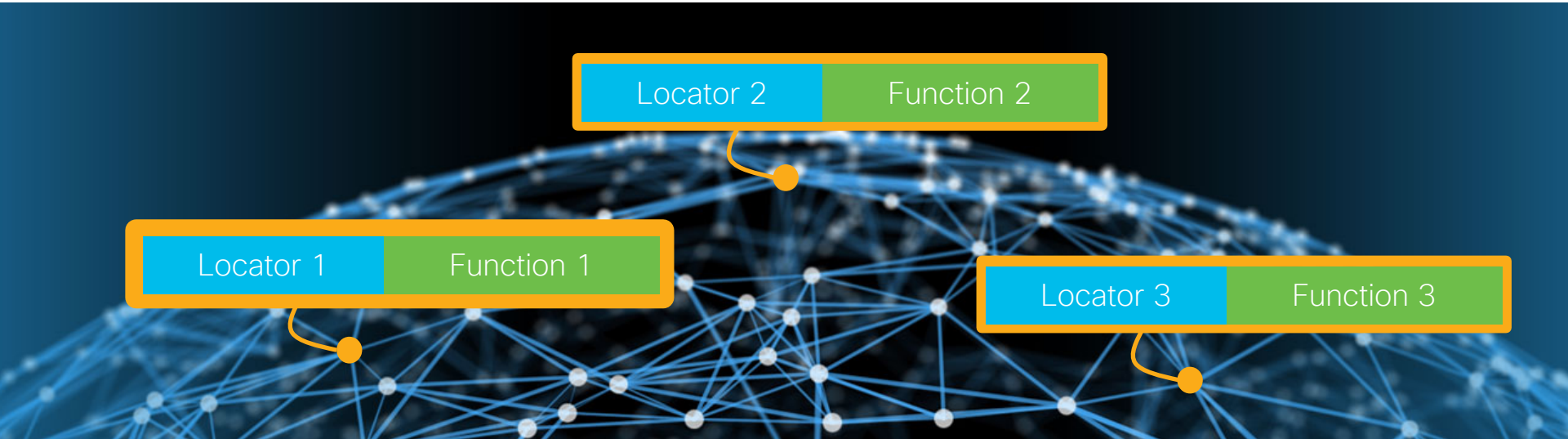
Network instruction

- 128-bit SRv6 SID
  - Locator: routed to the node performing the function
  - Function: any possible function
    either local to NPU or app in VM/Container
  - **Arguments: optional argument bits to be used only by that SID**
  - Flexible bit-length selection

Network Program

Next Segment →

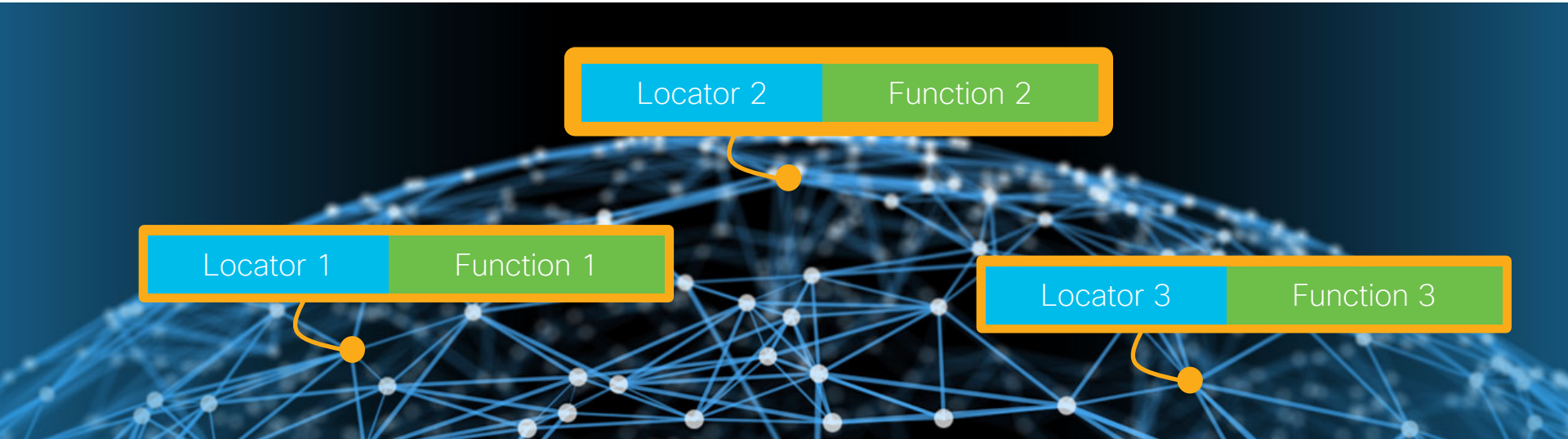
Locator 1	Function 1
Locator 2	Function 2
Locator 3	Function 3



Network Program

Next Segment

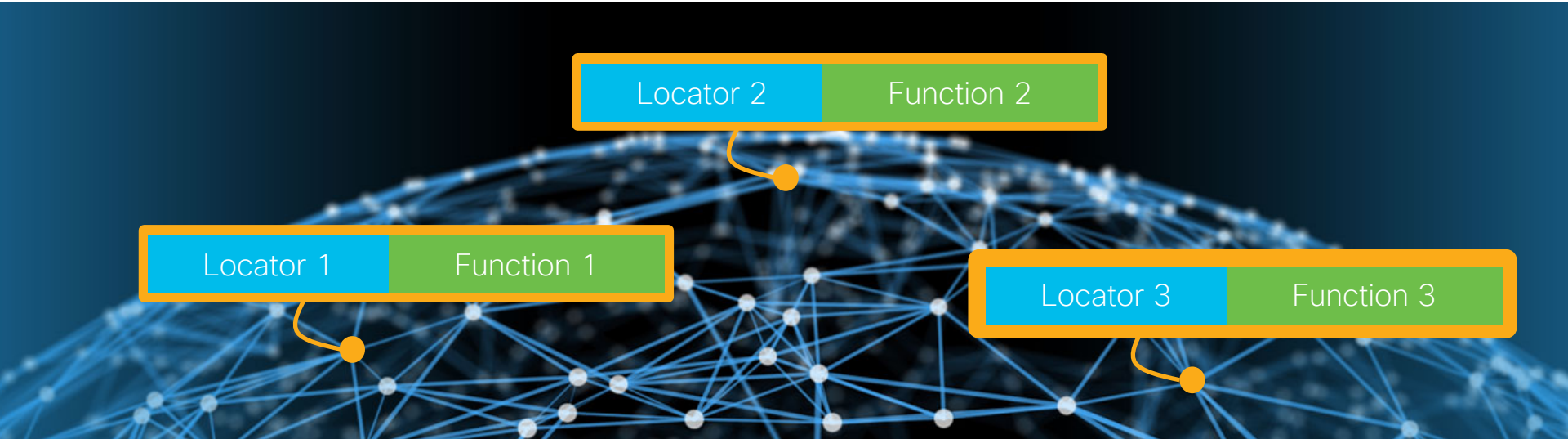
Locator 1	Function 1
Locator 2	Function 2
Locator 3	Function 3



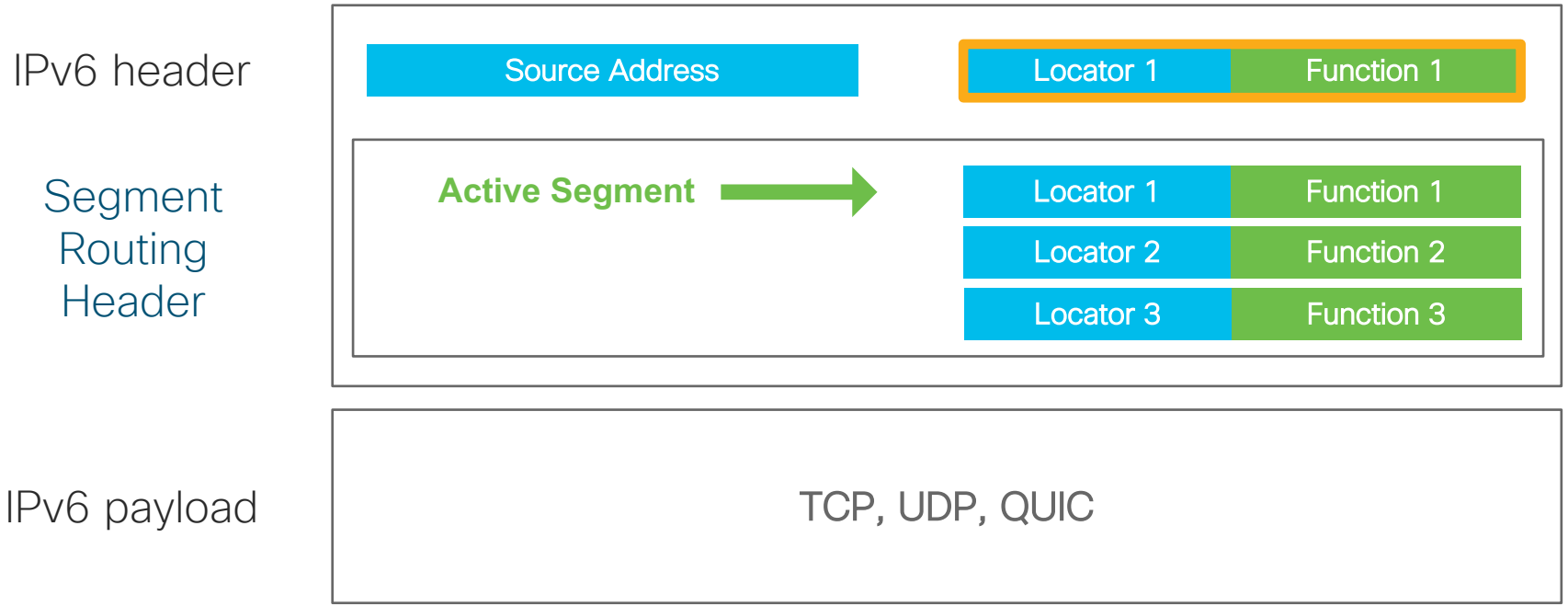
Network Program

Locator 1	Function 1
Locator 2	Function 2
Locator 3	Function 3

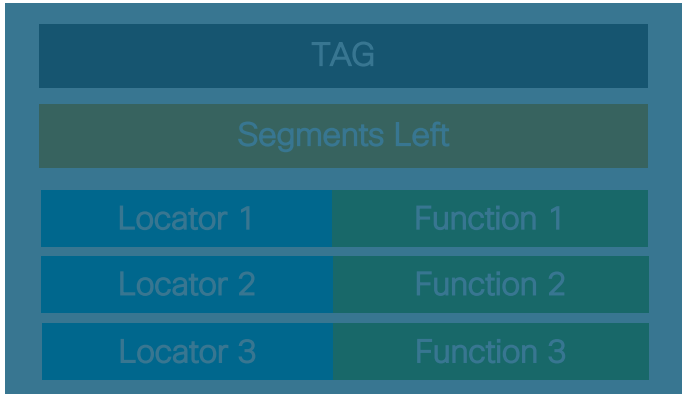
Next Segment →



Network Program in the Packet Header

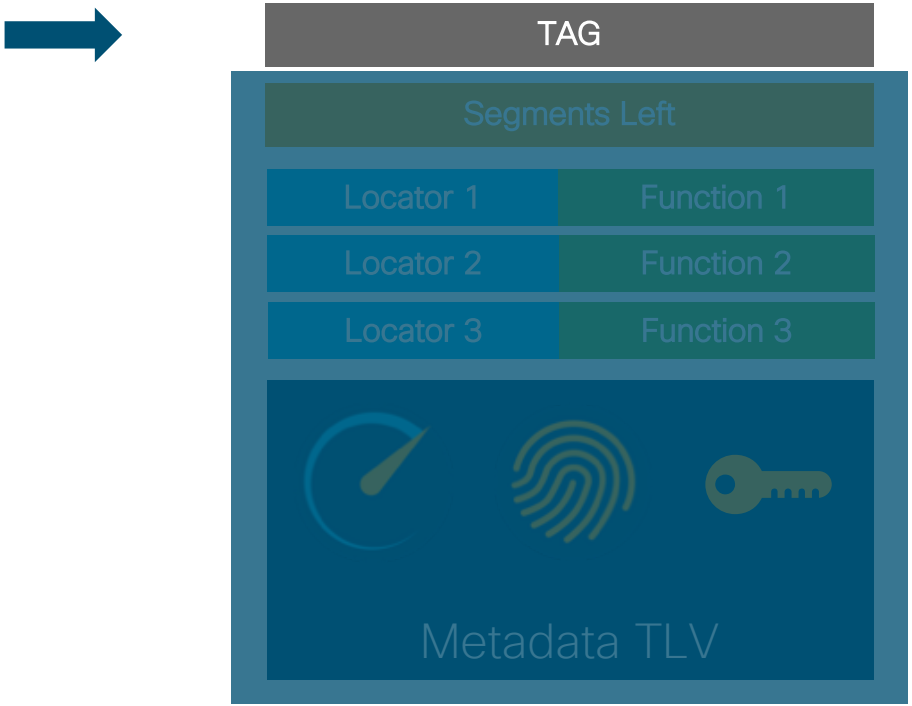


Argument shared between functions

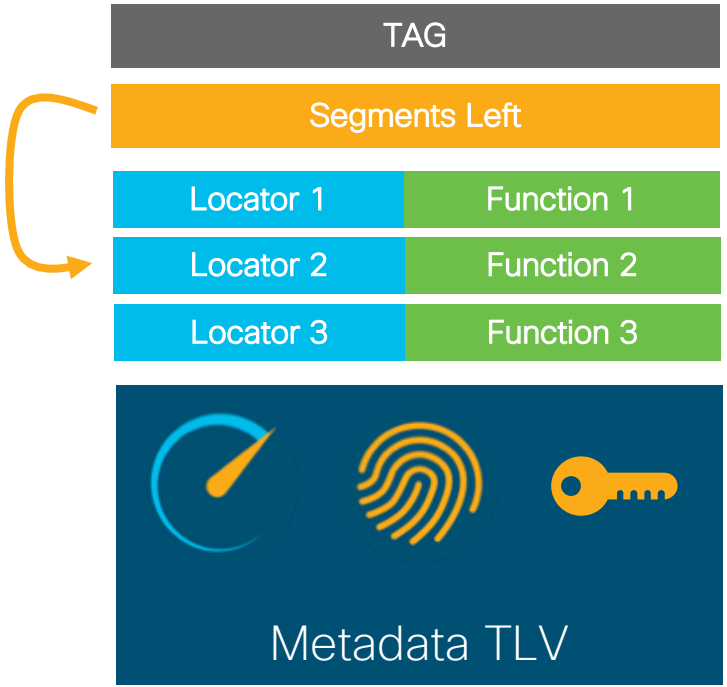


“Global” Argument

Group-Based Policy

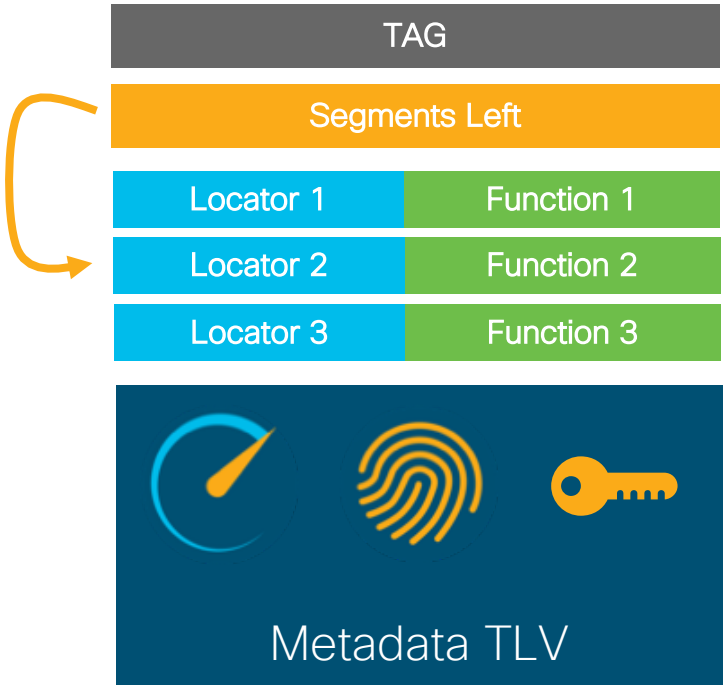


SRv6 Header




```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Next Header   |  Hdr Ext Len  | Routing Type  | Segments Left |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Last Entry   |     Flags     |              Tag              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Segment List[0] (128 bits IPv6 address)            |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                                                               |
                              ...
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|            Segment List[n] (128 bits IPv6 address)            |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                                                             //
//         Optional Type Length Value objects (variable)       //
//                                                             //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

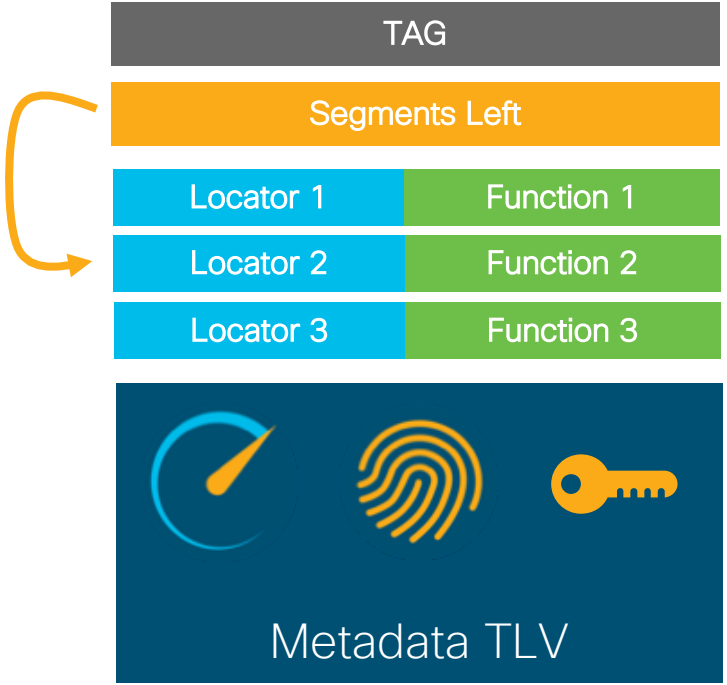
SRv6 for anything

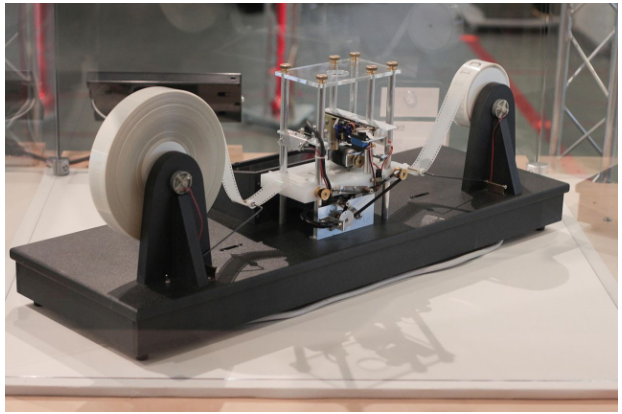


Optimized for HW processing
e.g. Underlay & Tenant use-cases

Optimized for SW processing
e.g. NFV, Container, Micro-Service

SRv6 for anything





Turing

Lead Operators

- Standardization
- Multi-Vendor Consensus

SPRING
Internet-Draft
Intended status: Standards Track
Expires: September 5, 2018

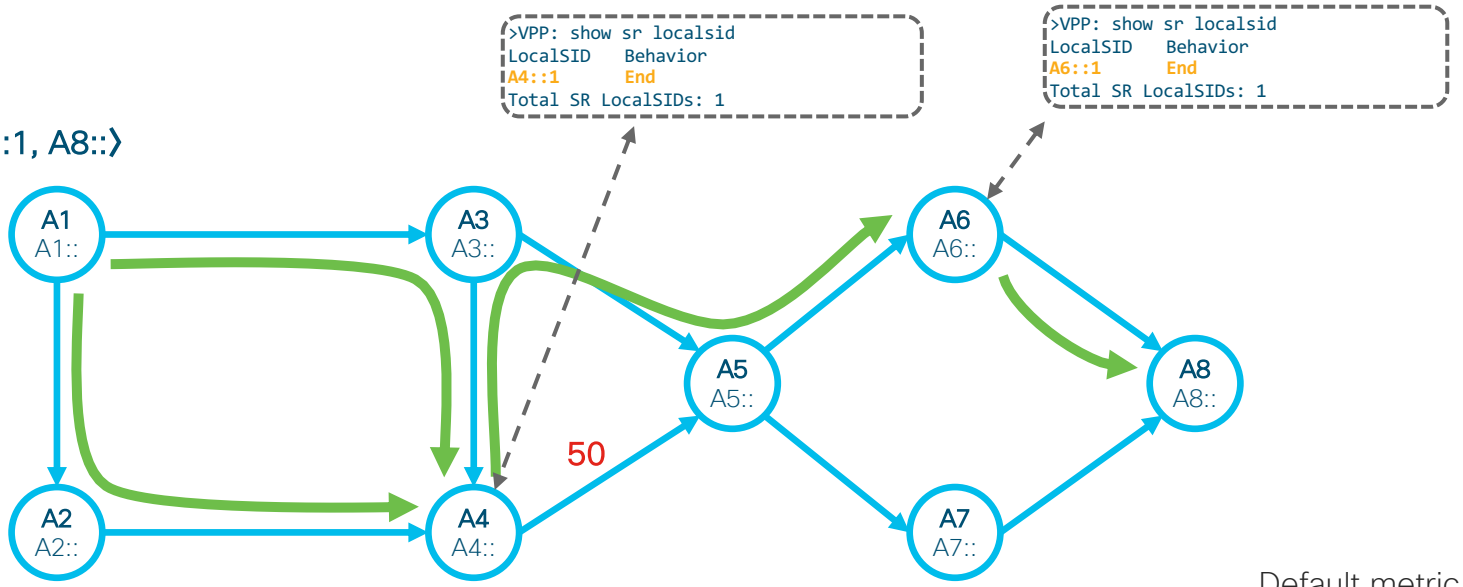
C. Filsfils
Cisco Systems, Inc.
Z. Li
Huawei Technologies
J. Leddy
Comcast
D. Voyer
D. Bernier
Bell Canada
D. Steinberg
Steinberg Consulting
R. Raszuk
Bloomberg LP
S. Matsushima
SoftBank
D. Lebrun
Universite catholique de Louvain
B. Decraene
Orange
B. Peirens
Proximus
S. Salsano
Universita di Roma "Tor Vergata"
G. Naik
Drexel University
H. Elmalky
Ericsson
P. Jonnalagadda
M. Sharif
Barefoot Networks
A. Ayyangar
Arista
S. Mynam
Innovium Inc.
W. Henderickx
Nokia
S. Ma
Juniper
A. Bashandy
K. Raza
D. Dukes
F. Clad
P. Camarillo, Ed.
Cisco Systems, Inc.
March 4, 2018

SRv6 Network Programming
draft-filsfils-spring-srv6-network-programming-04

SRv6 LocalSIDs

Endpoint function

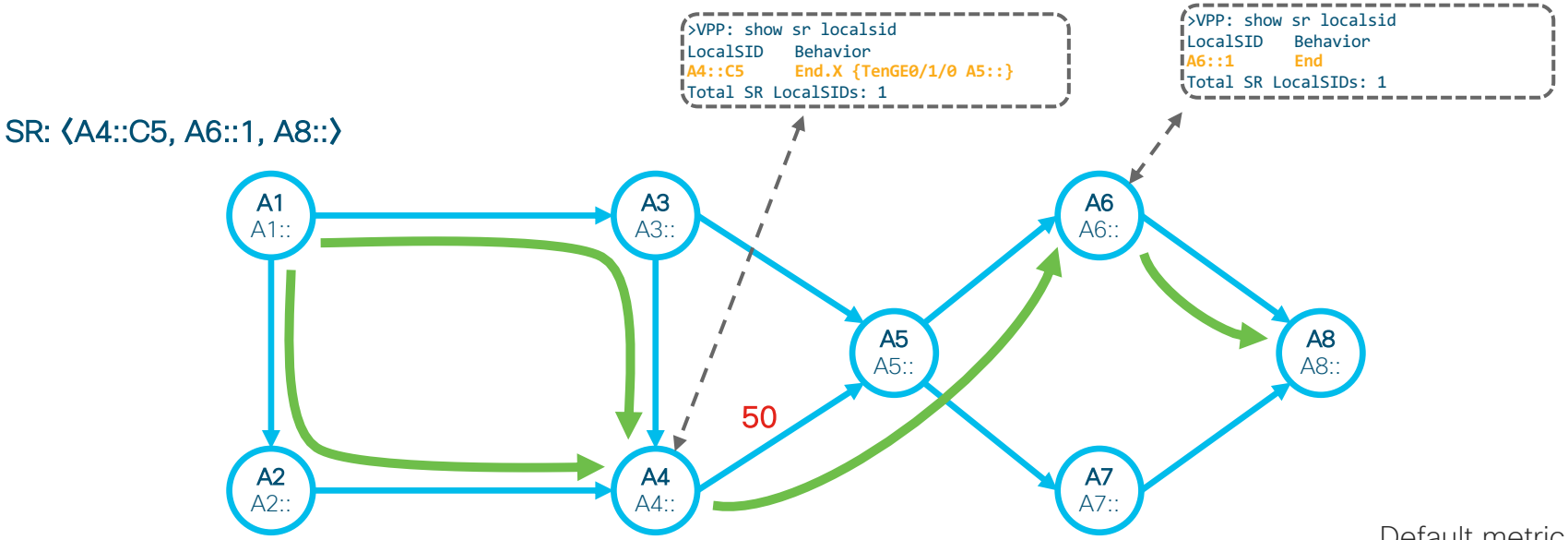
SR: ⟨A4::1, A6::1, A8::⟩



Default metric 10

- For simplicity function 1 denotes the most basic function
- Shortest-path to the Node

Endpoint then xconnect to neighbor function

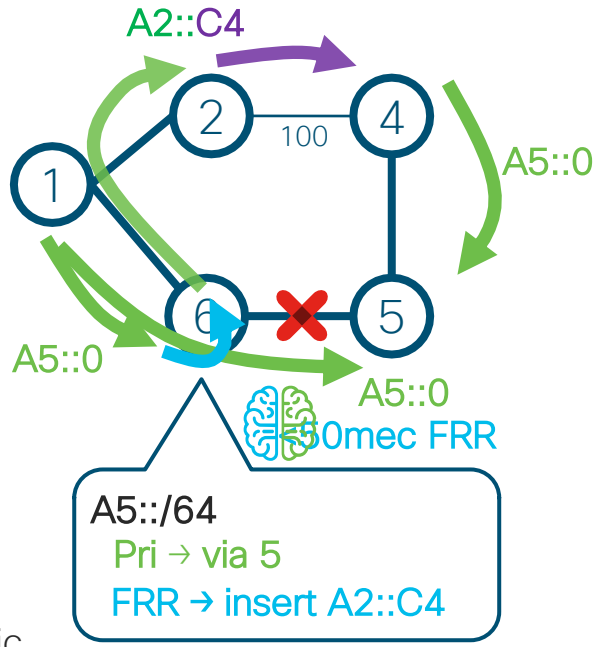


- For simplicity Ak::Cj denotes:
  - Shortest-path to the Node K and then x-connect (function C) to the neighbor J

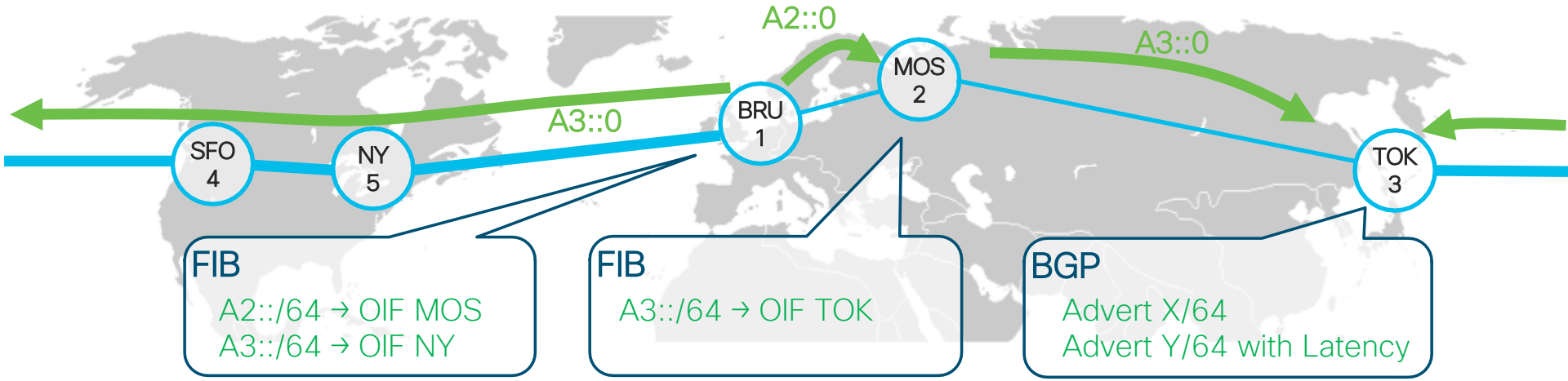
Deployment use-cases

TILFA

- 50msec Protection upon
  local link, node or SRLG failure
- Simple to operate and understand
  - automatically computed by the router's IGP process
  - 100% coverage across any topology
  - predictable (backup = postconvergence)
- Optimum backup path
  - leverages the post-convergence path, planned to carry the traffic
  - avoid any intermediate flap via alternate path
- Incremental deployment
- Distributed and Automated Intelligence

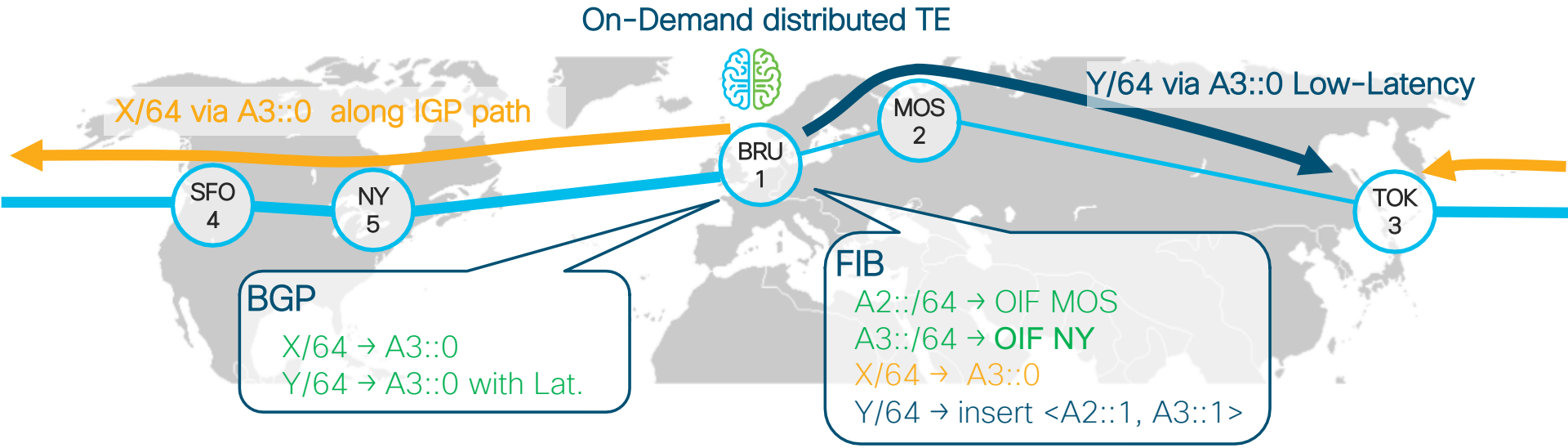


Distributed & Automated TE



- IGP minimizes cost instead of latency

Distributed & Automated TE



- Distributed and Automated Intelligence
- Dynamic SRTE Policy triggered by learning a BGP route with SLA contract
- No PBR steering complexity, No PBR performance tax, No RSVP, No tunnel to configure

Centralized TE

Input Acquisition

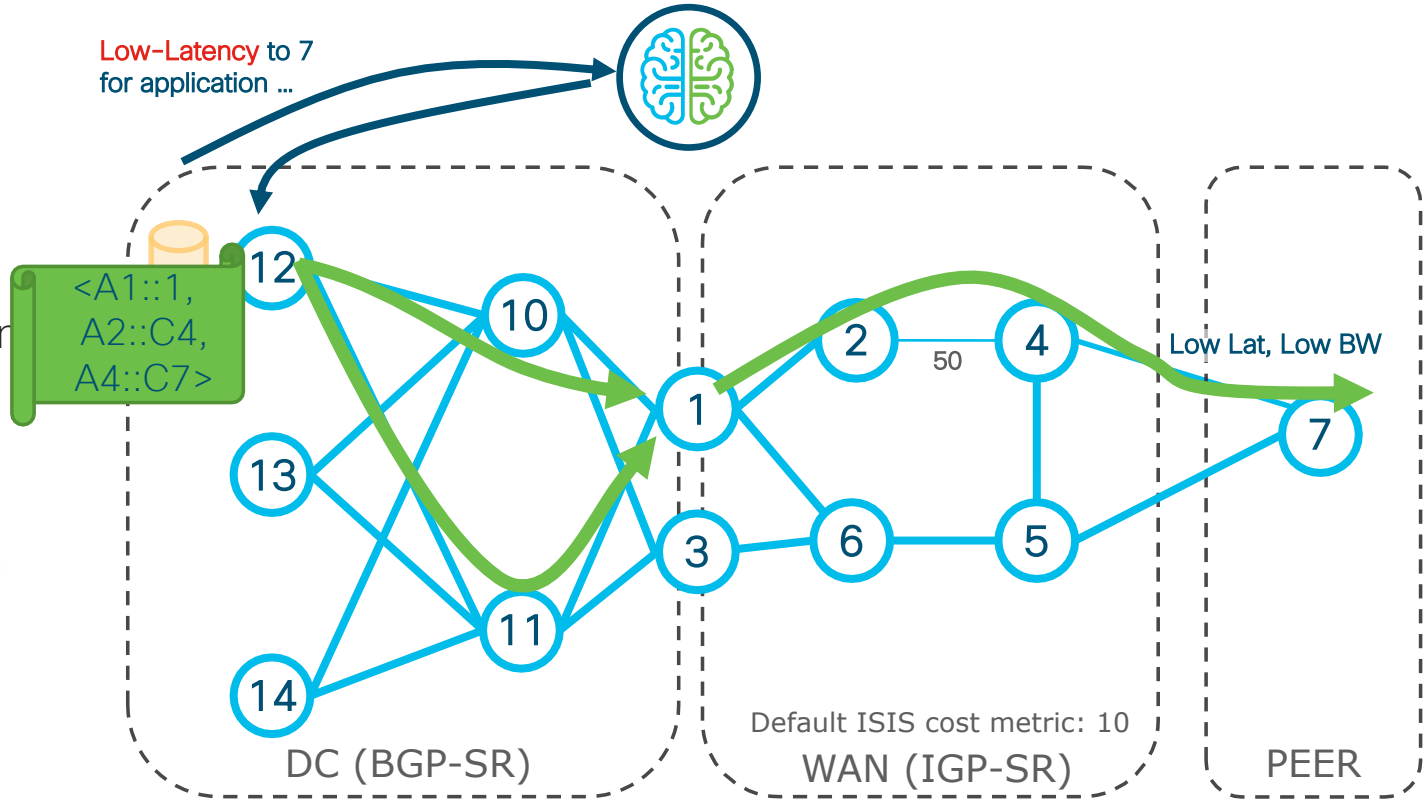
- BGP-LS
- Telemetry

Policy Instantiation

- PCEP
- BGP-TE
- Netconf / Yang

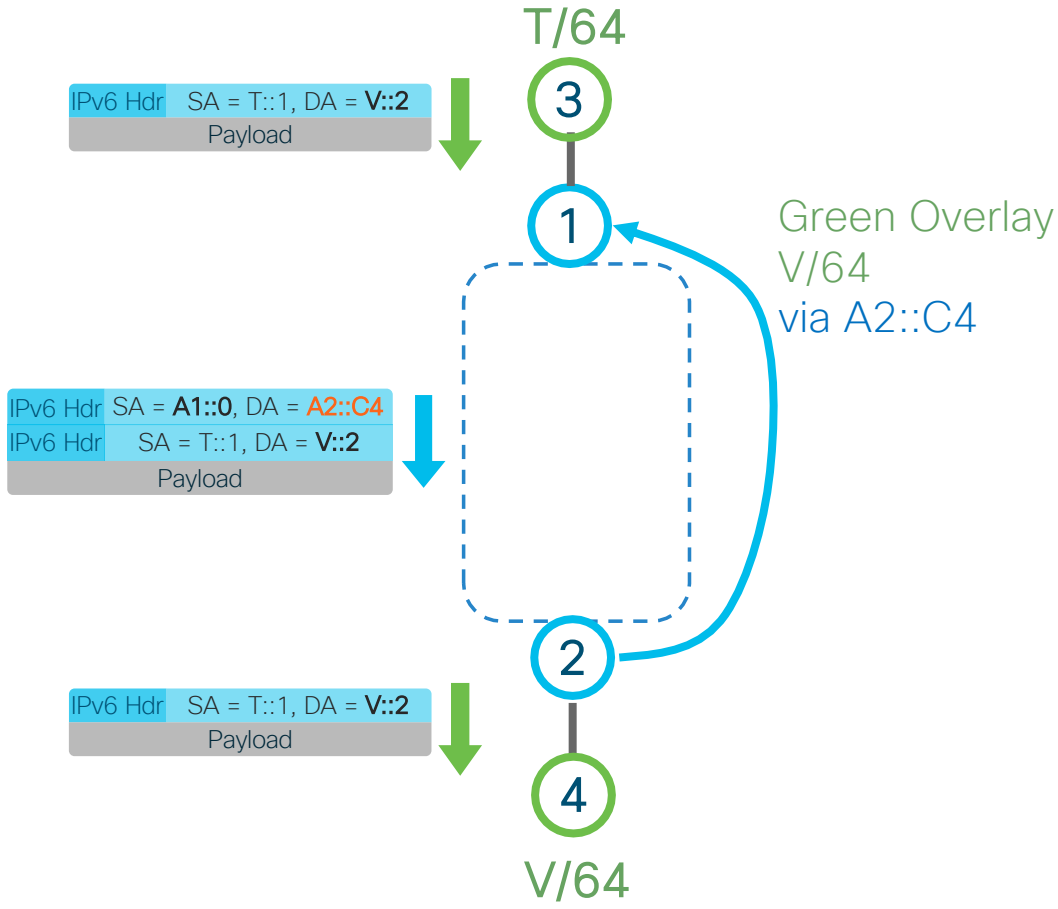
Algorithm

- SR native



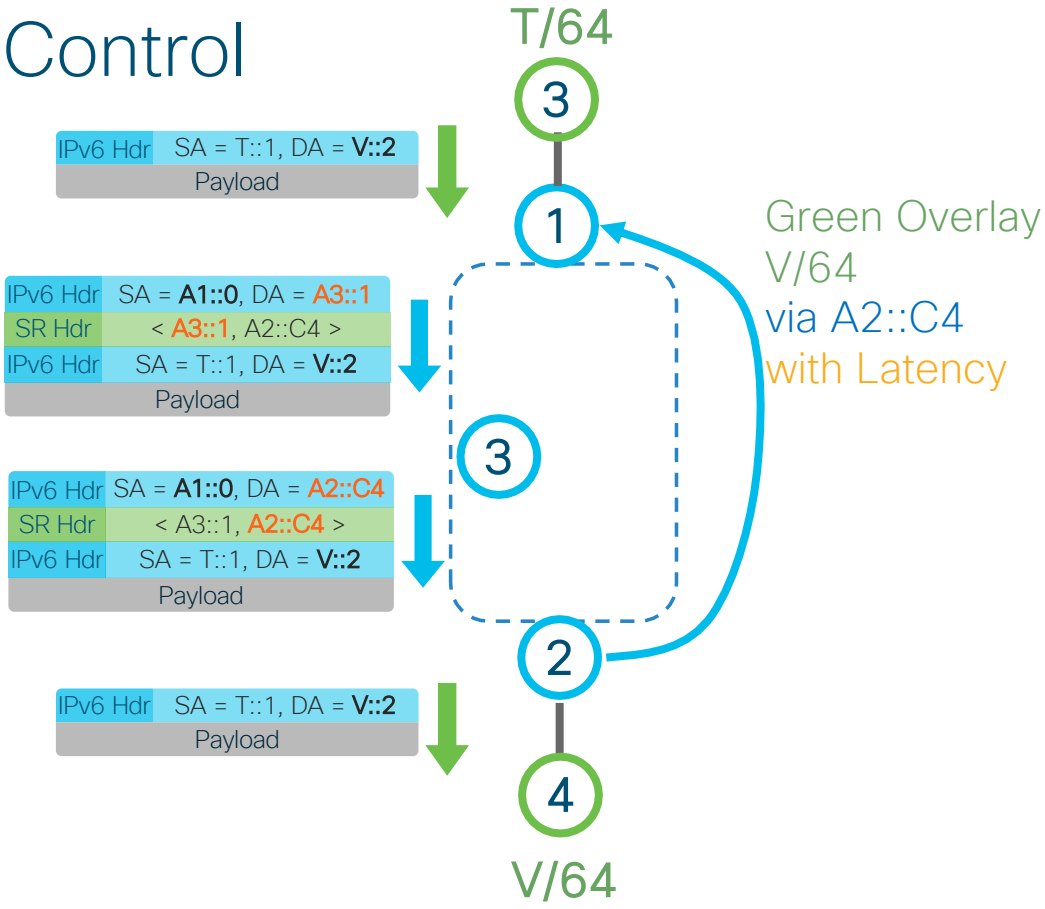
Overlay

- Automated
  - No tunnel to configure
- Simple
  - Protocol elimination
- Efficient
  - SRv6 for everything



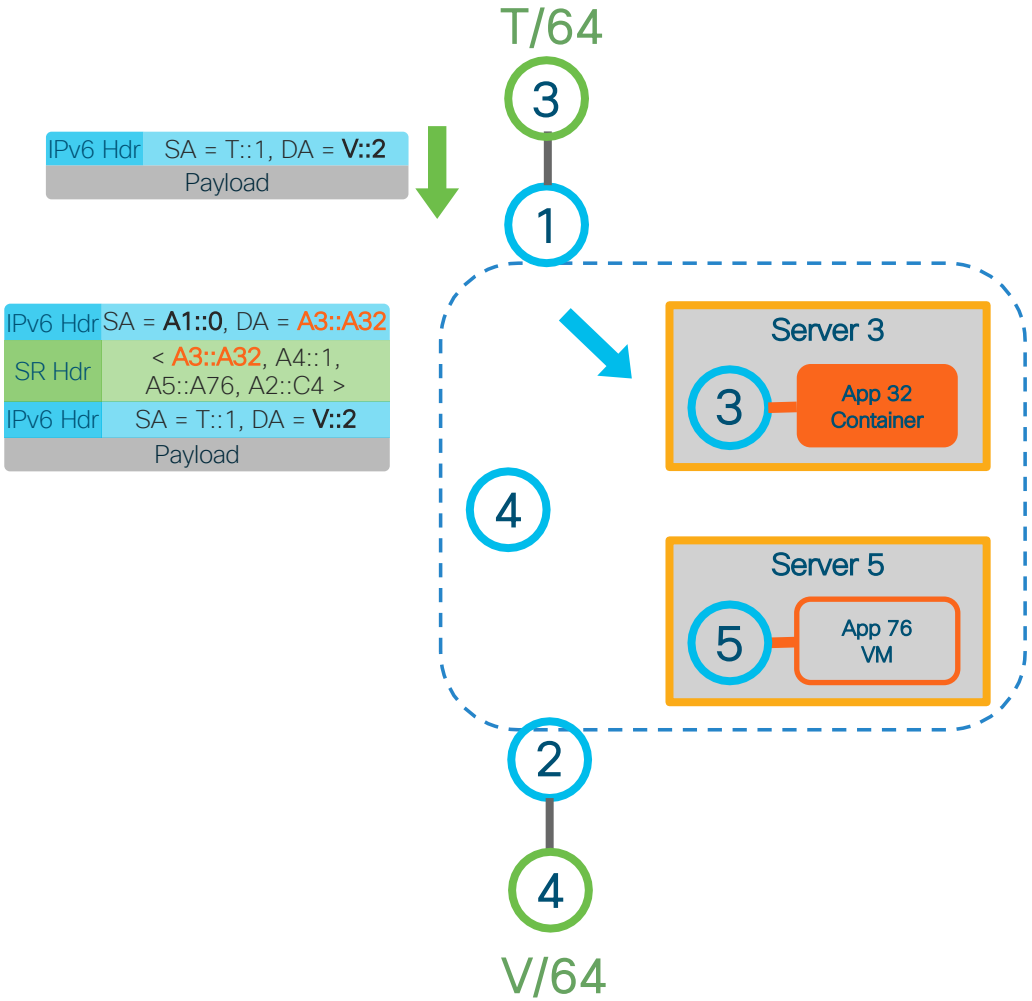
Overlay with Underlay Control

- SRv6 does not only eliminate unneeded overlay protocols
- SRv6 solves problems that these protocols cannot solve



Integrated NFV

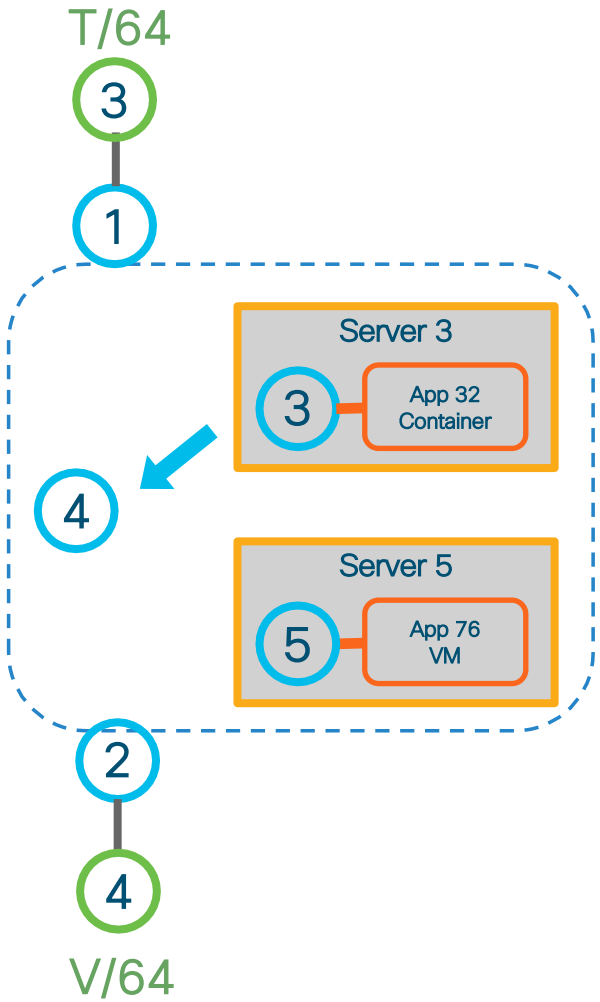
- Stateless
  - NSH creates per-chain state in the fabric
  - SR does not
- App is SR aware or not
- App can work on IPv4, IPv6 or L2



Integrated NFV

- Integrated with underlay SLA

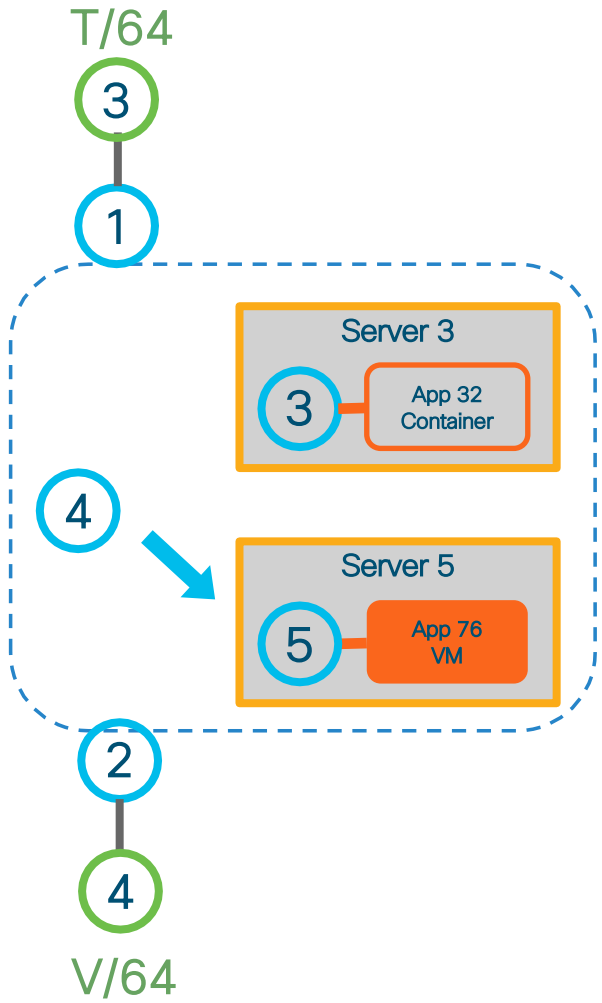
IPv6 Hdr	SA = **A1::0**, DA = **A4::1**
SR Hdr	< A3::A32, **A4::1**, A5::A76, A2::C4 >
IPv6 Hdr	SA = T::1, DA = **V::2**
Payload	



Integrated NFV

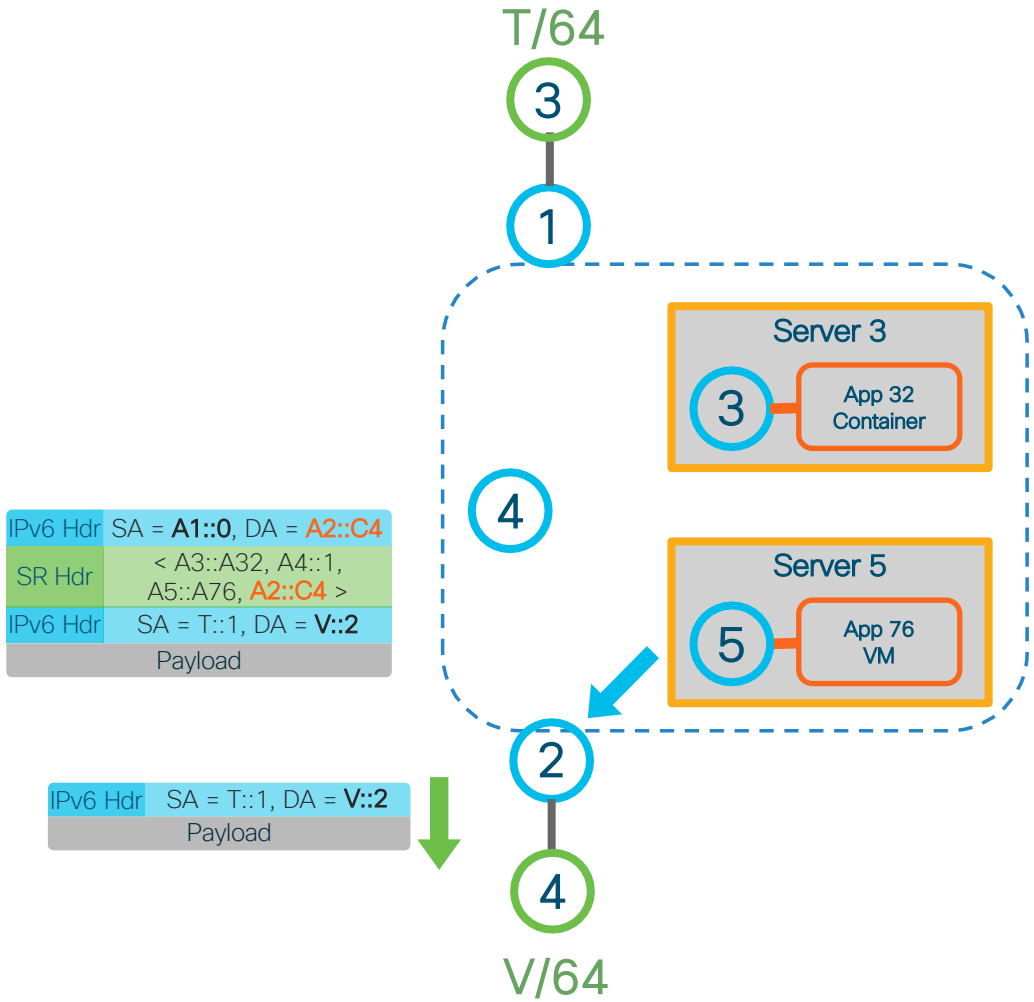
- Stateless
  - NSH creates per-chain state in the fabric
  - SR does not
- App is SR aware or not
- App can work on IPv4, IPv6 or L2

IPv6 Hdr	SA = **A1::0**, DA = **A5::A76**
SR Hdr	< A3::A32, A4::1, **A5::A76**, A2::C4 >
IPv6 Hdr	SA = T::1, DA = **V::2**
Payload	



Integrated NFV

- Integrated with Overlay



Service programming

Service Programming

Packets from are steered through a sequence of services on their way to the server

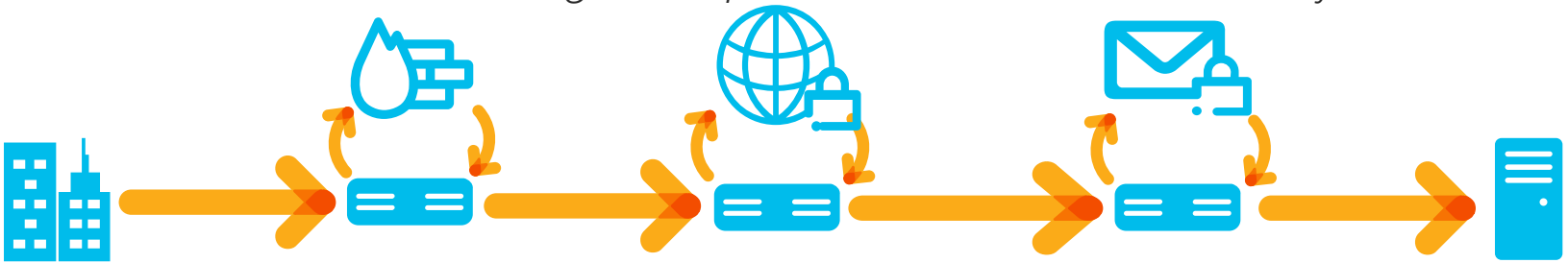
Service Programming – traditional approach

Packets from are steered through a sequence of services on their way to the server

- Services are placed on the traffic route
  - Static configurations
  - Traffic bottlenecks

Service Programming with **NSH**

Packets from are steered through a sequence of services on their way to the server



- Dedicated encapsulation header
  - State to be maintained for each service chain

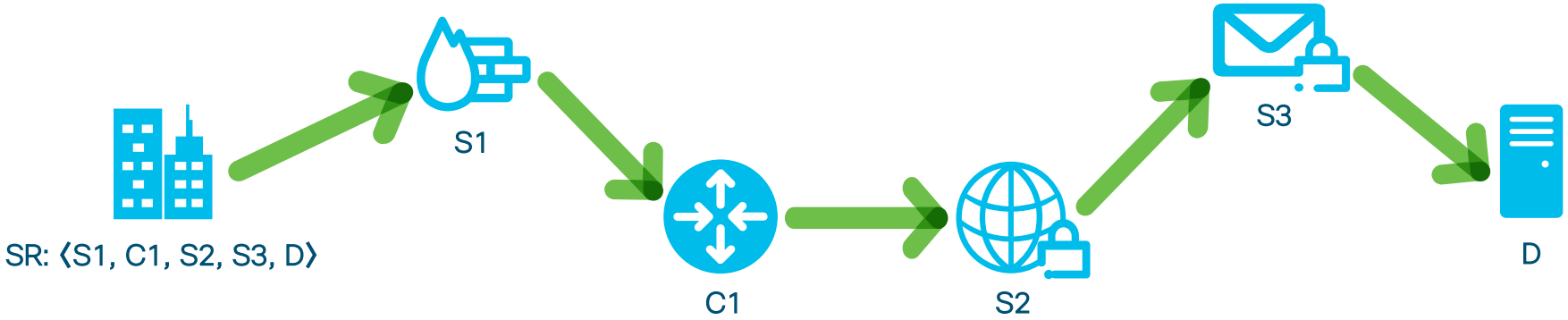
Service Programming with **SRv6**

Packets from are steered through a sequence of services on their way to the server

- **Services** are expressed with **segments**
  - Flexible
  - Scalable
  - Stateless

Service Programming with **SRv6**

Packets from are steered through a sequence of services on their way to the server



- **Services** are expressed with **segments**
  - Flexible
  - Scalable
  - Stateless

Service Programming with **SRv6**

Types of VNFs

SR-Aware VNFs:

- Leverage SRv6 Kernel support to create smarter applications
- SERA: SR-Aware Firewall (extension to iptables)

SR-UnAware VNFs:

- Application is not aware of SR at all
- Leverage VPP as a vm/container vSwitch to do SRv6 processing

SRv6 aware VNFs

- Leverage Linux Kernel 4.14 support for SRv6
- SERA: SR-aware firewall
  - Firewall rules based on the SRH
  - Firewall actions on the SRH
- Snort

Service Programming with **SRv6**

Types of VNFs

SR-Aware VNFs:

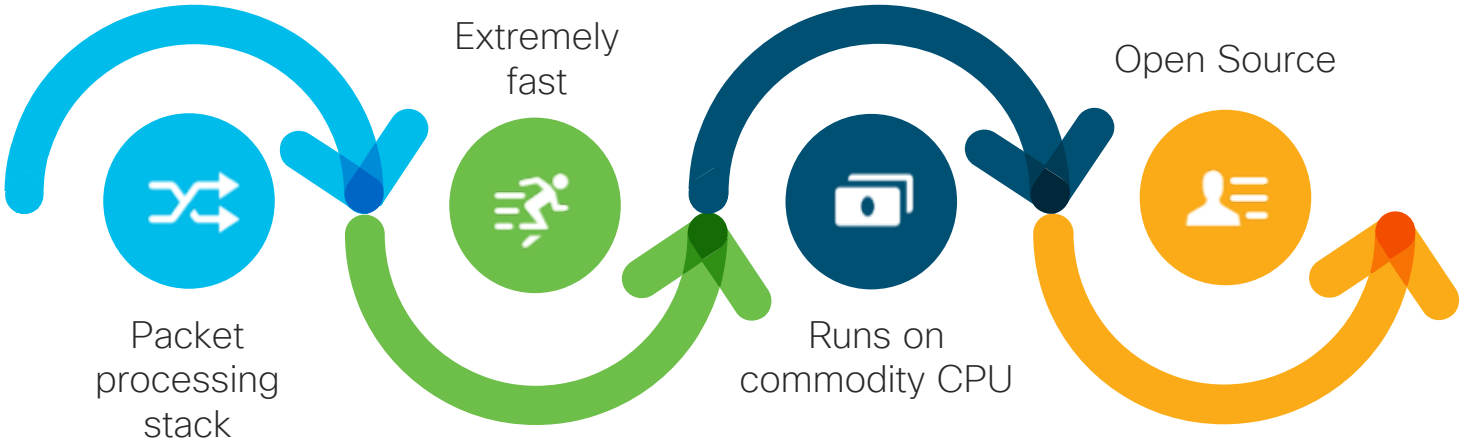
- Leverage SRv6 Kernel support to create smarter applications
- SERA: SR-Aware Firewall (extension to iptables)

SR-UnAware VNFs:

- Application is not aware of SR at all
- Leverage VPP as a vm/container vSwitch to do SRv6 processing

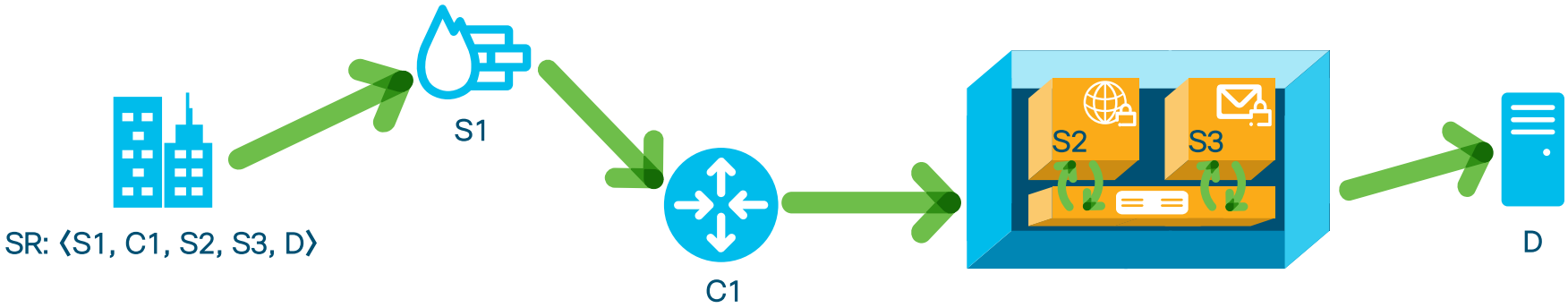
Vector Packet Processing

- Extensible framework that provides out-of-the-box production quality switch/router functionality (dataplane only)
- We've implemented the entire SRv6 Network Programming on it



SR-UnAware VNFs

- End.AM – Endpoint to SR-unaware app via masquerading
- End.AD – Endpoint to SR-unaware app via dynamic proxy
- End.ASM – Endpoint to SR-unaware app via shared memory



End.ASM – Endpoint to SR-unaware app via shared mem.

1. Put the received packet in a shared memory region
2. Perform SR processing on the host
   Pass a **pointer** of the inner packet to S2
3. Perform SR processing on the host
   Pass a **pointer** of the inner packet to S3
4. Move the packet from the shared memory into the output iface buffer ring

- Valid for IPv4 and IPv6 traffic
- Max. theoretical achievable performance

SR to the Host

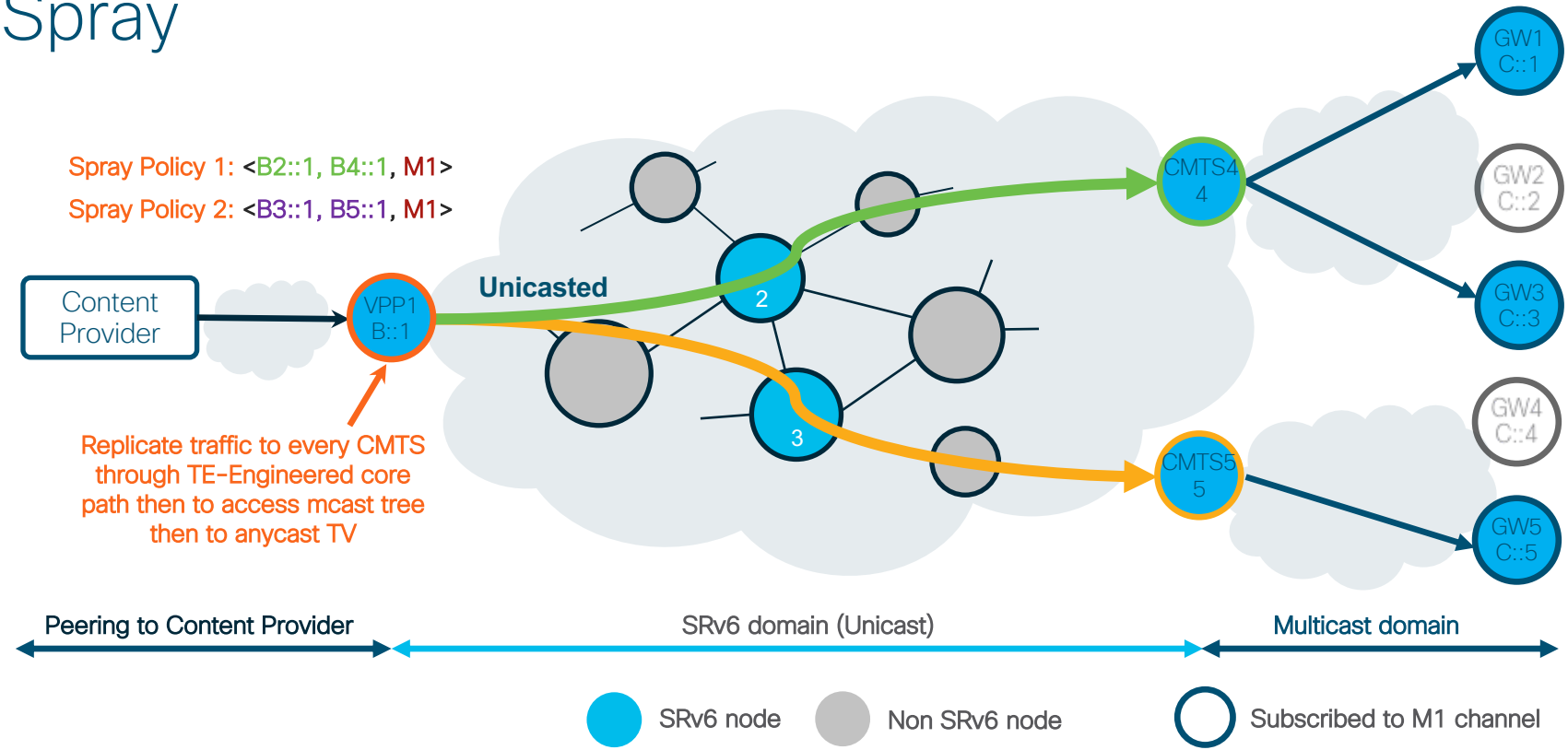
- Why Application Responsive Networking?
  - Revenue opportunities are moving towards the applications (hosted experiences, contextual experiences, etc)
  - Applications have no visibility over the network or mechanisms to request optimization objectives
  - IETF: Path Aware Networking RG (panrg)
    “bringing path awareness to transport and application layer protocols…”
  - Smarter applications allows to distribute function processing over the network’s edges
- Let’s rethink service programs policies
  - Leverage ”Loc::Fun:Arg” SRv6 SID format to embed function parameters
  - Leverage TLVs for complex metadata or in-band telemetry

	Locator	Function	Arguments	
Firewall with Policy Identifier	2605:A800:FFFE:1111:A100:	B1:	:0100	-> Policy ID
Rate-Limiting Policy	2605:A800:FFFE:1111:A100:	C1:	:1234	-> Threshold
Video transcoder	2605:A800:FFFE:1111:A100:	D1:	A15 : 273	-> Format/bitrate
JIT video packaging	2605:A800:FFFE:1111:A100:	F1:	A :0512	-> Package format

Agenda

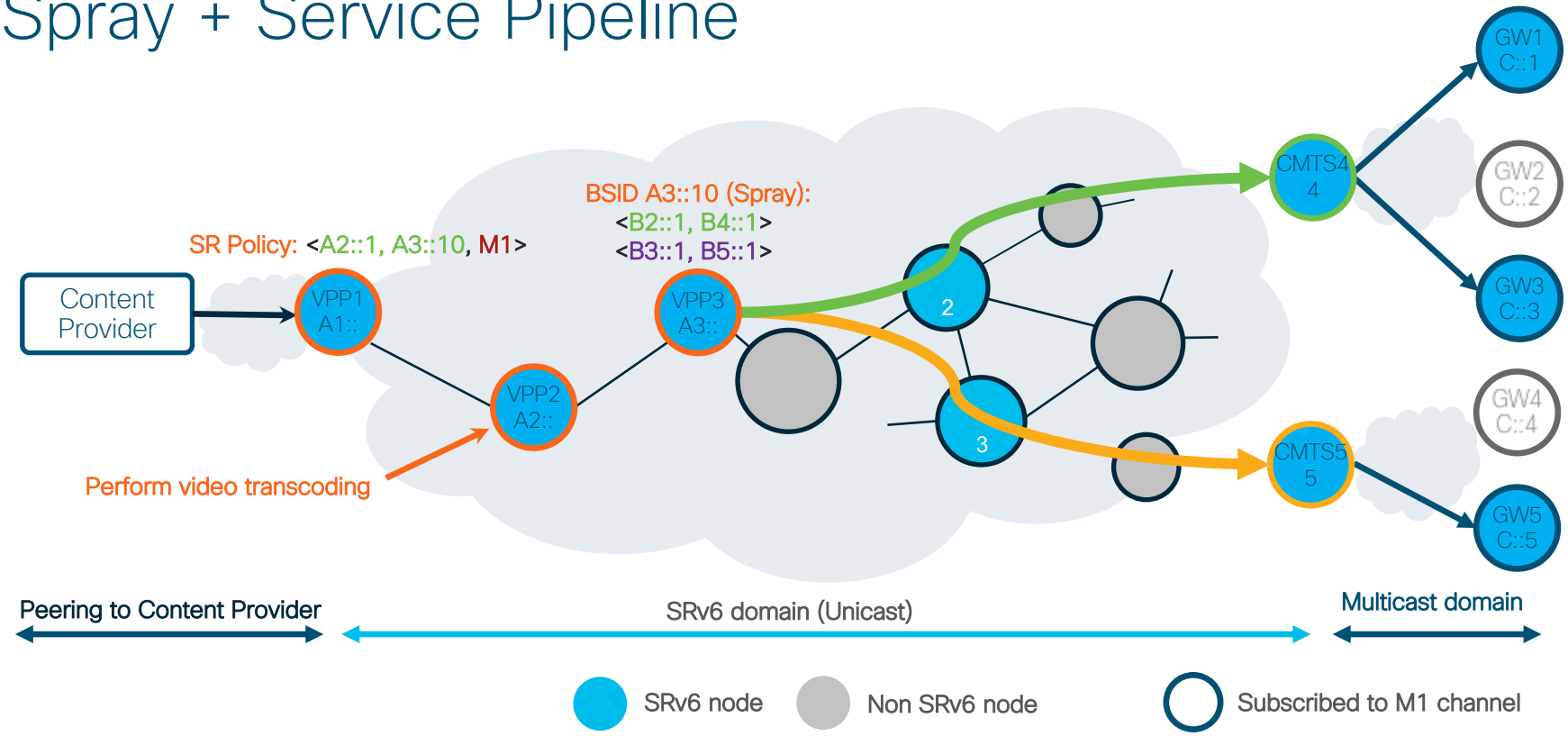
1. SRv6 101 
2. SRv6 LocalSIDs functions 
3. Deployment use-cases 
4. VPN Overlay 
5. Service Programming 
6. Spray
7. SD-WAN
8. 5G and network slicing

Spray



Flexible, SLA-enabled and efficient content injection without multicast core

Spray + Service Pipeline



Efficient distribution with flexible video processing

SD-WAN

Binding SID

- **A Binding SID is a unique ‘alias’ of an SR policy. ***
- If a packet arrives with the BSID, then the SR policy is applied on such packet
- Several Binding SIDs may point to the same SR policy

- Upon topology changes within the core of the network, the low-latency path may change. While the path of an intermediate policy changes, **its BSID does not change**.
- Provides scaling, network opacity and service independence.
- A BSID acts as a stable anchor point which isolates one domain from the churn of another domain.

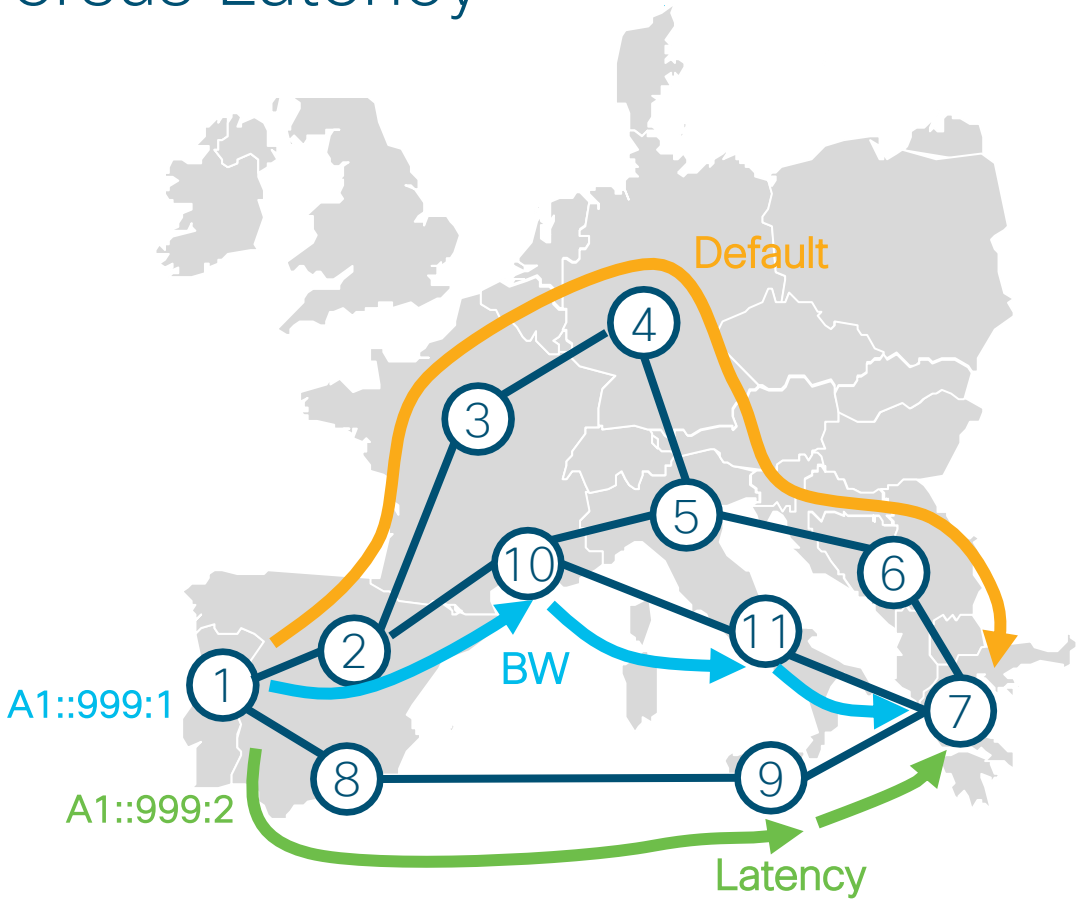
* Naïve definition of a BSID

SD-WAN

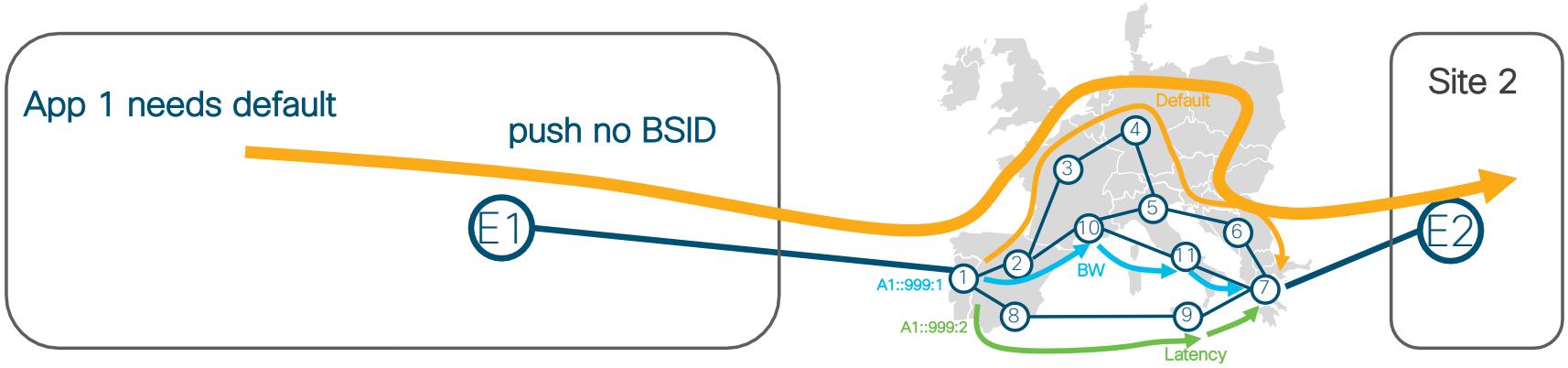
- Delegates the application recognition and policy decision to the Entreprise who knows better **when** an application needs a non-default path and **which** non-default path is needed
- NFV service chaining and Traffic-Engineering policies can be integrated in a SR policy
- Applicability to both SR-MPLS and SRv6
- To simplify, let's focus on
  - TE/SLA policy
  - SRv6

Default versus BW versus Latency

- Lisbon (1) to Athens (7)
- Default
  - <A7::>
- BW: Guaranteed 50Mbps
  - <A10::1, A11::1, A7::>
  - BSID: A1::999:1
- Low-Latency
  - <A9::1, A7::>
  - BSID: A1::999:2

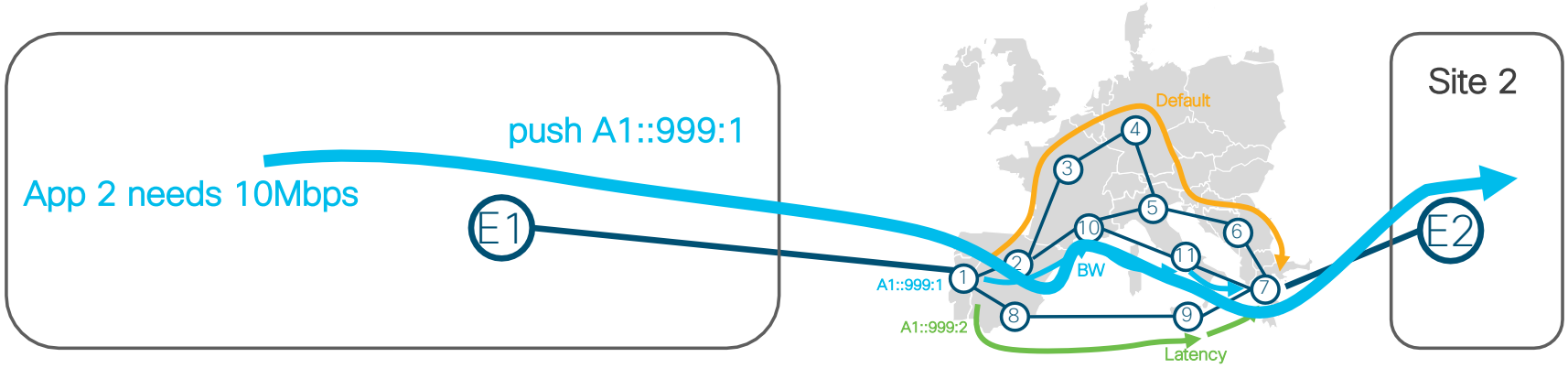


App needs best-effort



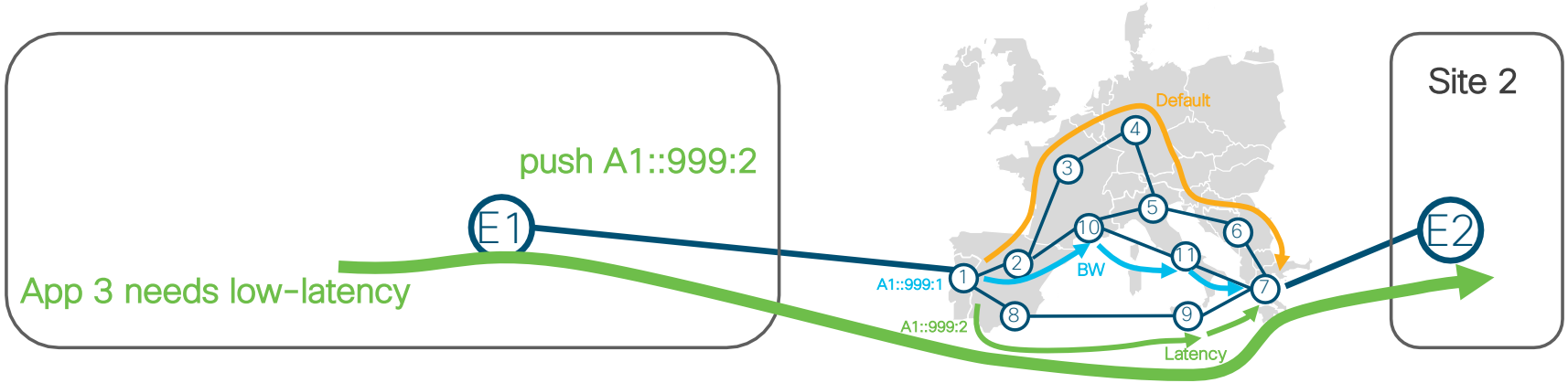
- E1 encrypts the inner packet and encapsulate in outer packet to E2
- E1 does not push any BSID

App needs guaranteed BW



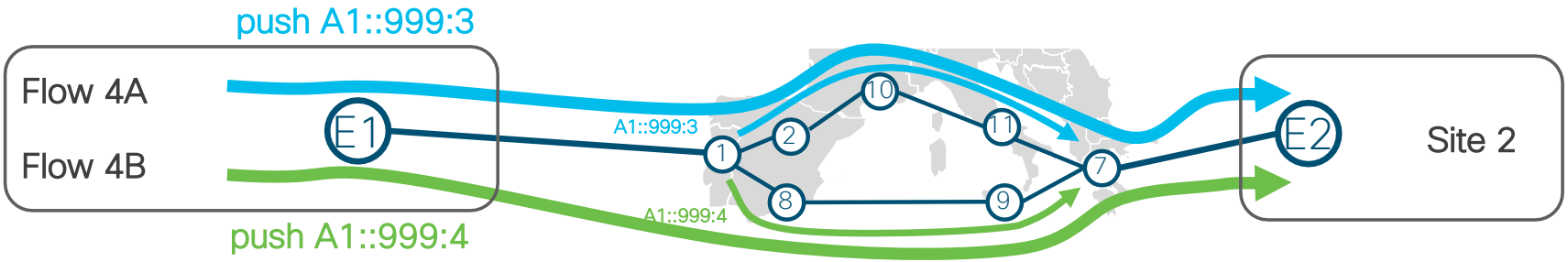
- E1 encrypts the inner packet and encapsulate in outer packet to E2
- E1 pushes A1::999:1
- The network provides the guaranteed BW service to App2

App needs low-latency



- E1 encrypts the inner packet and encapsulate in outer packet to E2
- E1 pushes A1::999:2
- The network provides the low-latency service to App3

Disjointness



- App 4 needs flow F4A and F4B to reach site 2 via disjoint paths
- E1 encrypts the inner packets and encapsulate in outer packet to E2
- For F4A, E1 additionally pushes A1::999:3
- For F4B, E1 additionally pushes A1::999:4

Binding SID is crucial in SD-WAN

- Identifier for a customized SLA per application per Entreprise
- Secured
- Per-BSID counters for usage-based billing

- Delegates the application recognition and policy decision to the Entreprise who knows better **when** an application needs a non-default path and **which** non-default path is needed

Performance monitoring

- Enterprise-based
  - Enterprise can easily monitors each individual service
  - Simply sends the probes with the related BSID
- Service Provider-based
  - The SP can enable per-SR-policy performance monitoring (latency/loss)
  - These metrics can be leveraged by SDWAN controller and provided to the Enterprise
    - BSID Metadata to select which application to steer

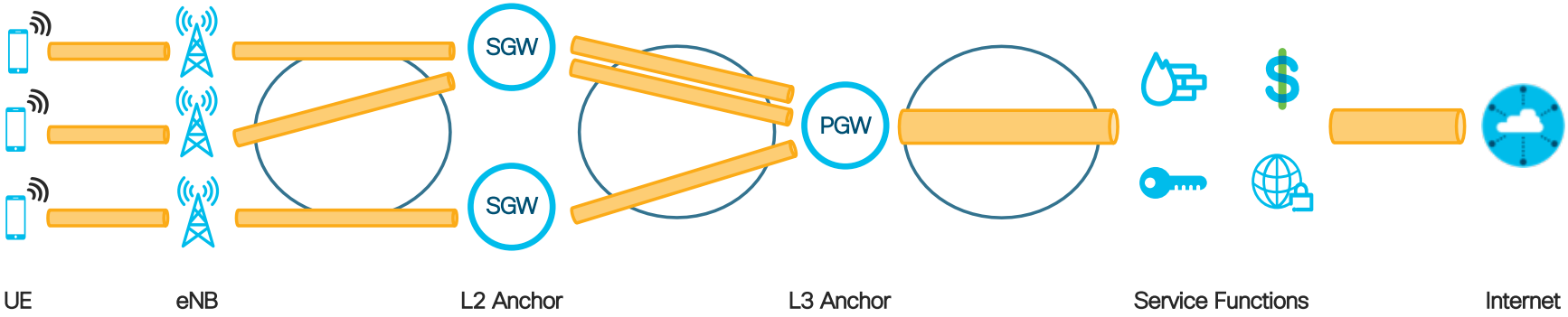
5G and Network Slicing

Current mobility networks

- Well fragmented RAN, EPC, SGi
  - Inefficient data paths
  - Protocol stack gets large
- Per-session tunnel creation
- Per-mobility event tunnel handling

Does **not** scale to 5G requirements:

- Increased number of connected devices
- Ultra-low latency
- Network slicing
- Mobile edge computing



SRv6 for mobile user-plane

- What about if SRv6 becomes an alternative to GTP-U?
  - Removing the per-session tunneling has obvious benefits
  - Optimal data path (ultra-low latency)
  - Integrated service chaining (allows for NFVs for security, billing, ...)

- **Native** support for **network slicing**
  - Achieved either via a **centralized SDN solution** or via **SR TE with IGP FlexAlg**
  - Optimal resource utilization

- Well-progressed standardization
  - IETF: draft-ietf-dmm-srv6-mobile-uplane-01
  - 3GPP: Accepted study item in CT4 (#29.892)

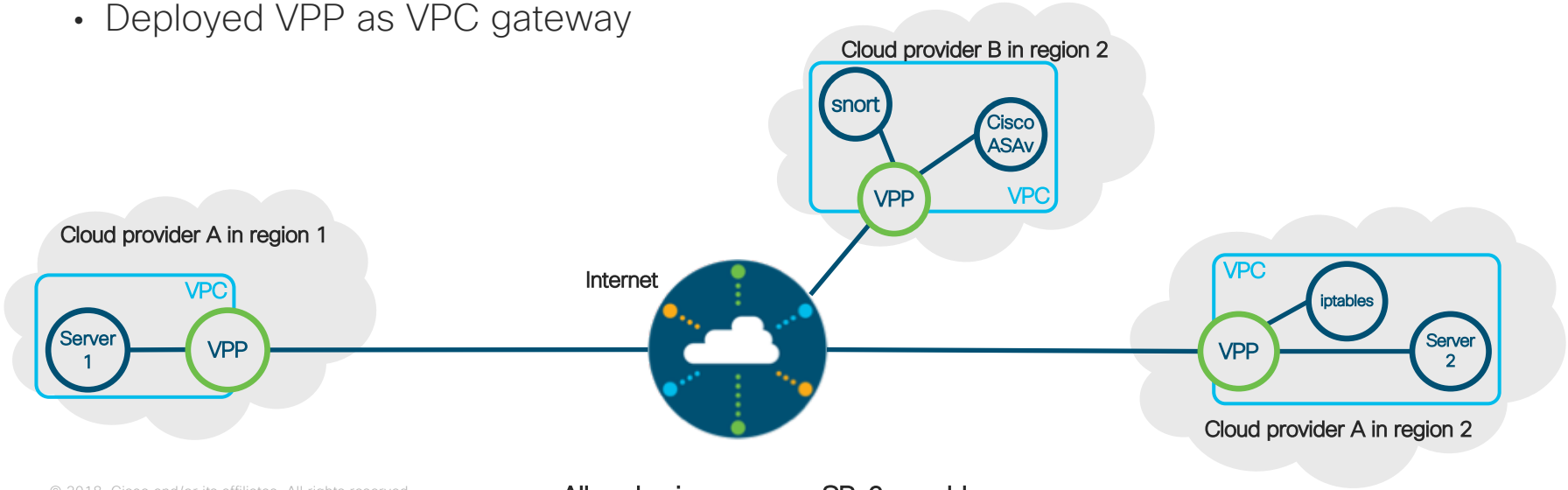
SRv6 for mobile user-plane

- SRv6 offers –in an integrated manner-:
  - Overlay
  - Underlay
  - Service chaining
- Draft focuses on a slow migration path for N9 and N3 interfaces
  - Traditional mode
  - Enhanced mode
  - Interworking mechanisms
- SRv6 is a stepping-stone for newer control-planes that might come latter

Multi-cloud overlays

Multi-cloud overlays

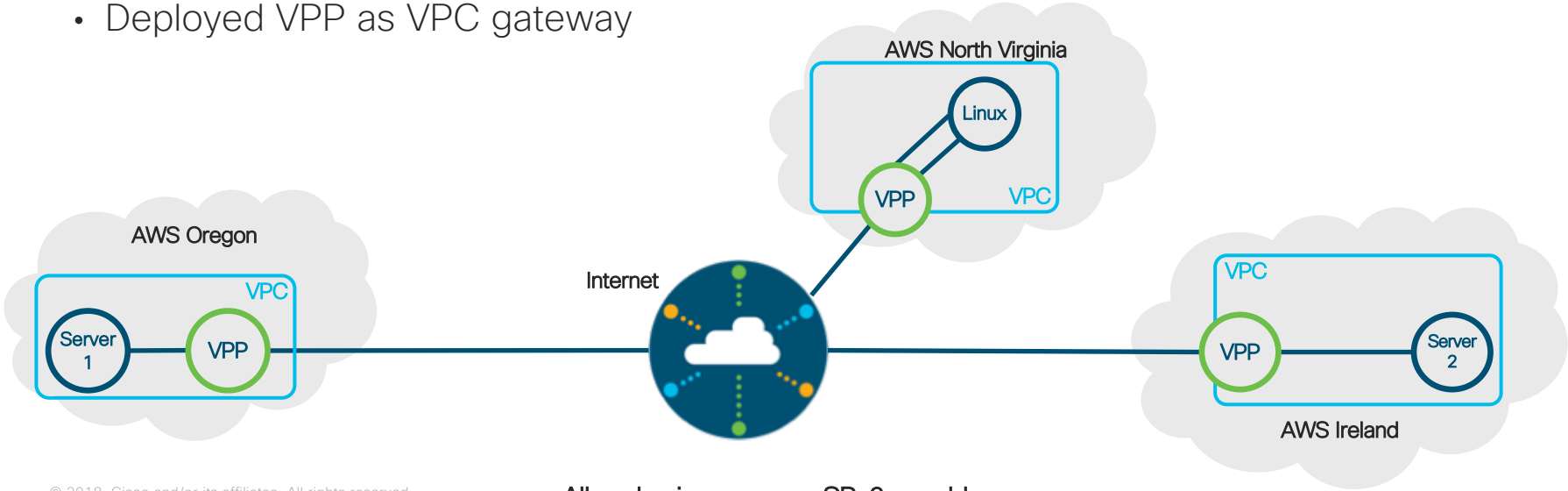
- How do you **interconnect** several **cloud-provider regions** (as an **end-customer**)?
- Transit is **plain IPv6** which we **do not control**
- Let's use **SRv6** for the **overlay** and **service chaining** only
- Deployed VPP as VPC gateway



All nodes in green are SRv6 capable

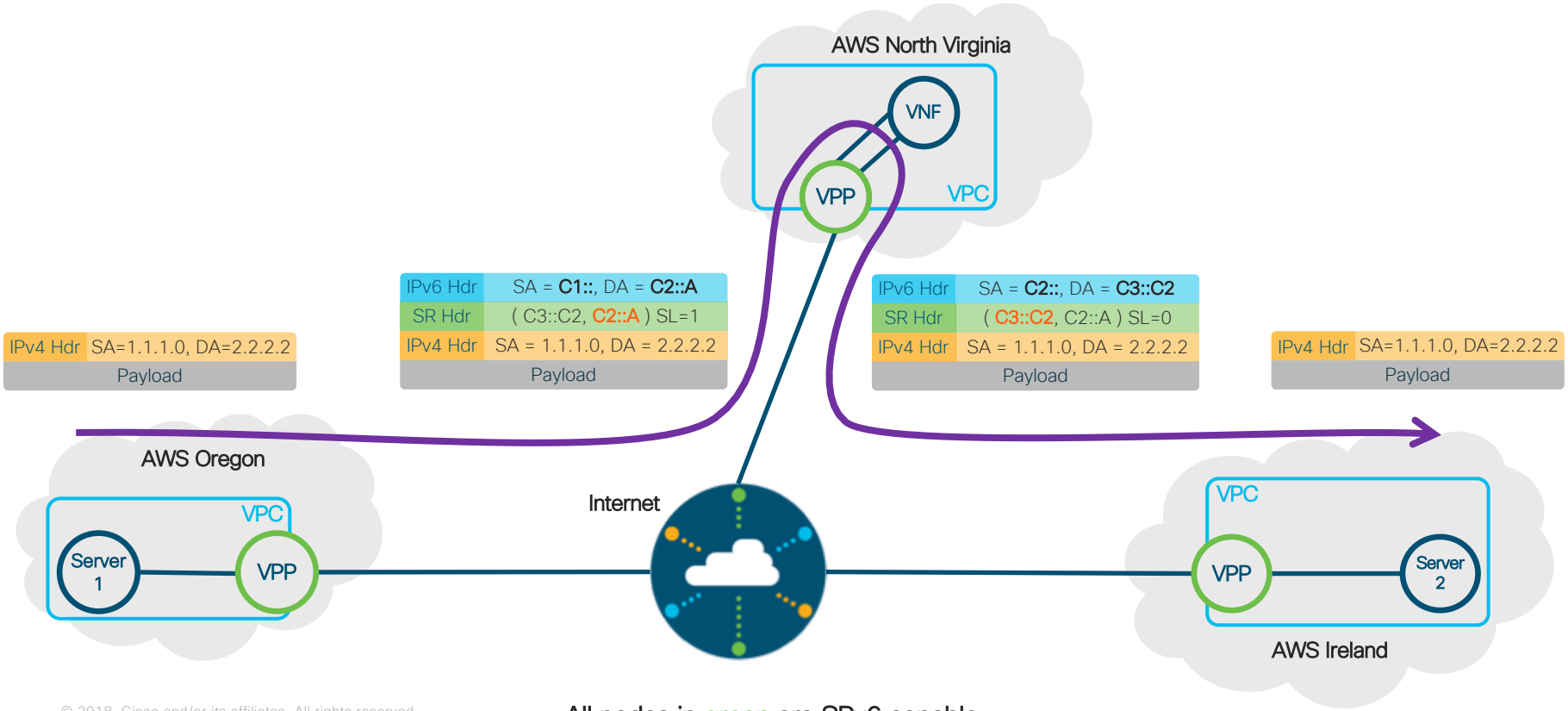
Multi-cloud overlays

- How do you **interconnect** several **cloud-provider regions** (as an **end-customer**)?
- Transit is **plain IPv6** which we **do not control**
- Let's use **SRv6** for the **overlay** and **service chaining** only
- Deployed VPP as VPC gateway



All nodes in green are SRv6 capable

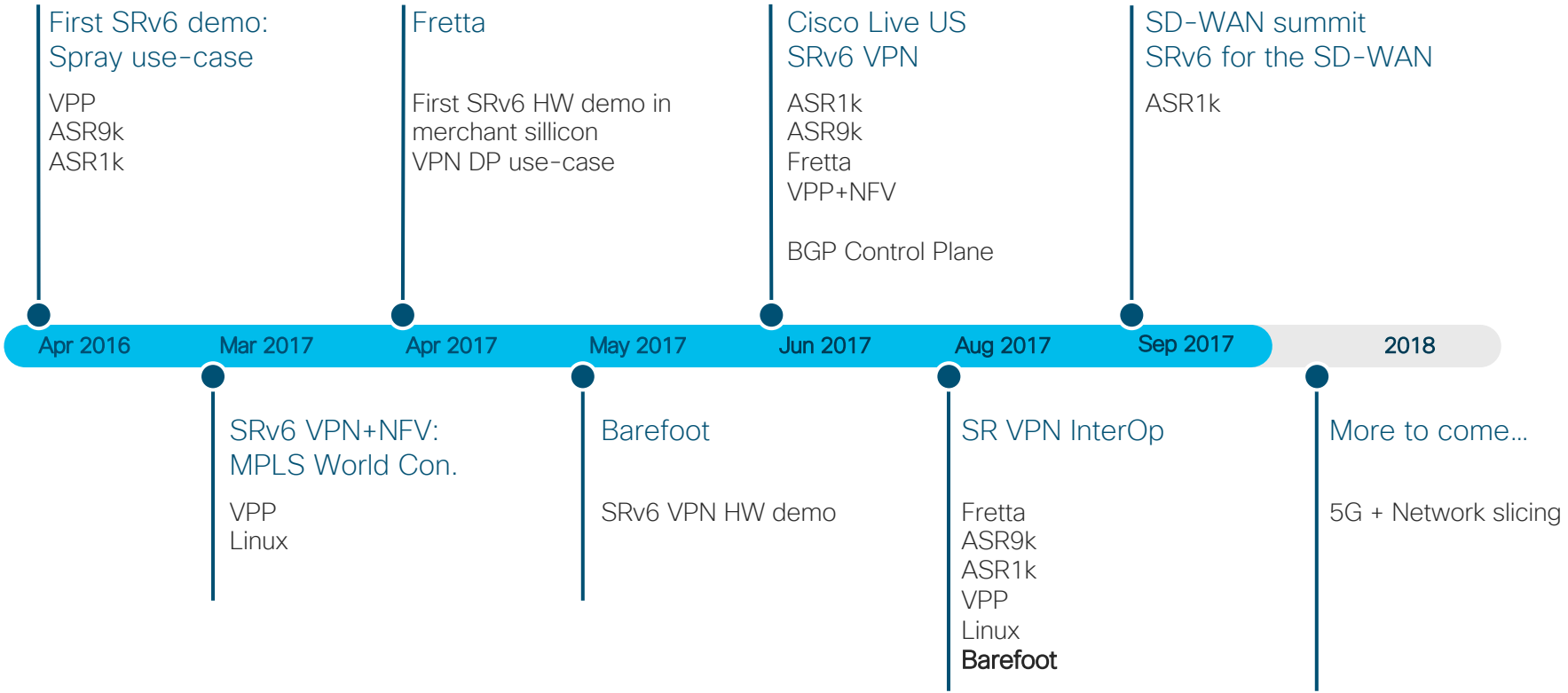
Multi-cloud overlays



All nodes in green are SRv6 capable

Where are we?

SRv6 timeline

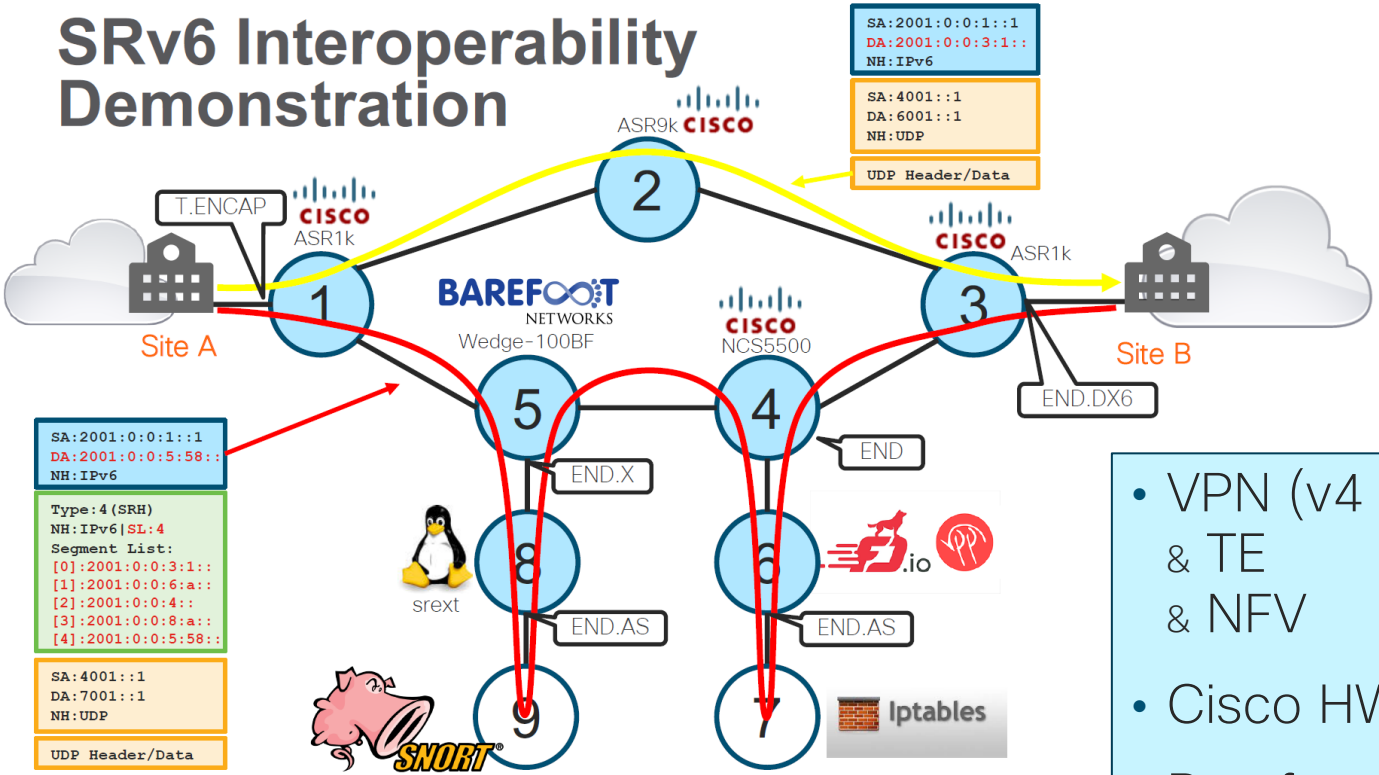


Implementations

- Cisco HW
  - NCS5k - XR
  - ASR9k - XR
  - ASR1k – XE
- Open-Source
  - Linux 4.10
  - FD.IO
- Barefoot HW

© 2018 Cisco and/or its affiliates. All rights reserved.

SRv6 Interoperability Demonstration



- VPN (v4 and v6)
  & TE
  & NFV
- Cisco HW with XR and XE
- Barefoot HW with P4 code
- FD.IO
- Linux

blogs.cisco.com/sp/segment-routing-ipv6-interoperability-demo-is-already-there

Conclusion

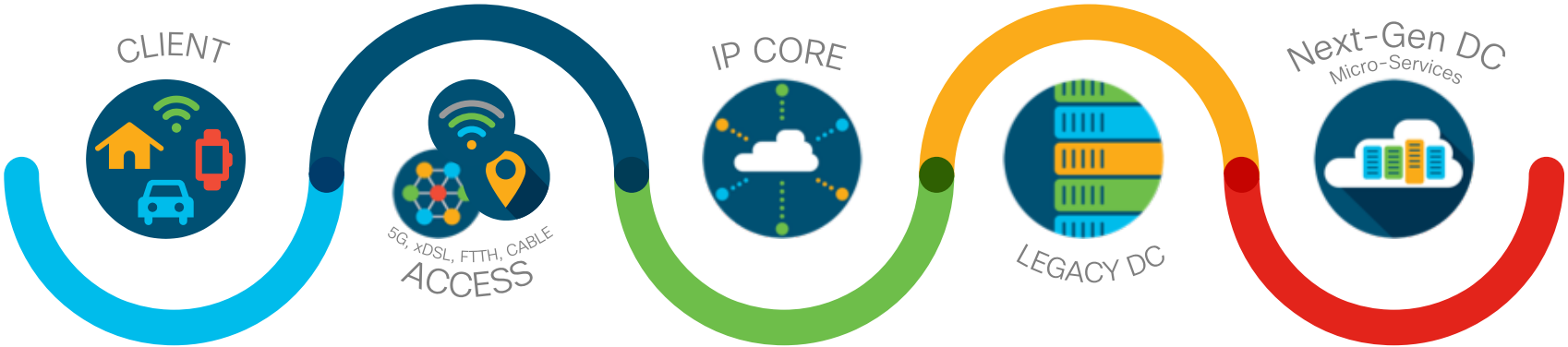
Segment Routing conclusion

- Strong industry support
- Fantastic deployment rate
- Bold architecture: network programming
- Numerous use-cases
  - FRR, TE, SDN, Overlay with SLA, NFV, Spray, SD-WAN, 5G & NS, ...
- Feel free to join the lead-operator team!

Partnering

- Track-record collaboration with operator
  - Focus on real operator needs
  - Seamless Deployment
  - Standardization
  - Multi-Vendor consensus
- Looking forward to working together

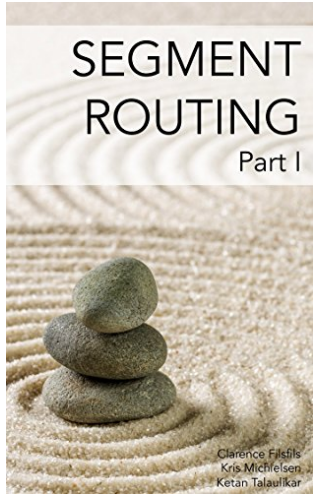
IPv6 provides reachability



SRv6 unleashes IPv6 potential

Stay up-to-date

amzn.com/B01I58LSUO



segment-routing.net

linkedin.com/groups/8266623

twitter.com/SegmentRouting

facebook.com/SegmentRouting/

Thank you!

pcamaril@cisco.com