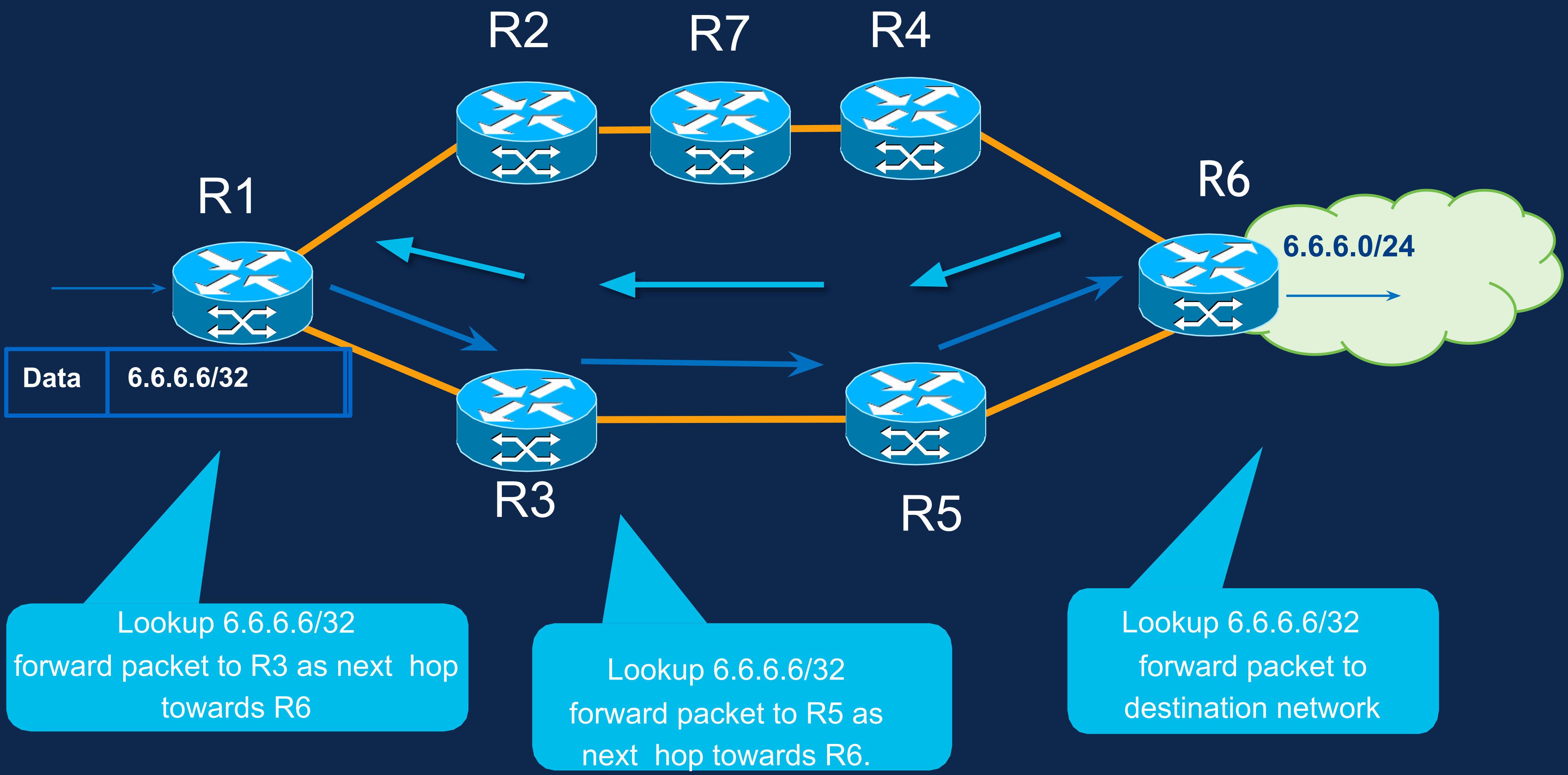
Demystifying Segment Routing (SR-MPLS)

AUSNOG 2022

Sanjeewa Alahakone, CCIE 8462

Cisco Systems

Routing Protocols, Destination Based Data Flow



MPLS Brief History

Original drivers towards label switching

- Designed to make router faster.
  - Routers were originally built to handle forwarding in the software on a single CPU.
  - ATM switches were faster than routers.
  - MPLS allows the device to do same job as router with performance of ATM switch.
- Late 1996, proprietary multilayer solutions emerged with integrated ATM switching and IP routing:
  - IP Switching—Ipsilon/Nokia
  - Tag Switching—Cisco Systems
  - Cell Switching Router (CSAggregate Route-Based IP Switching (ARIS)—IBM

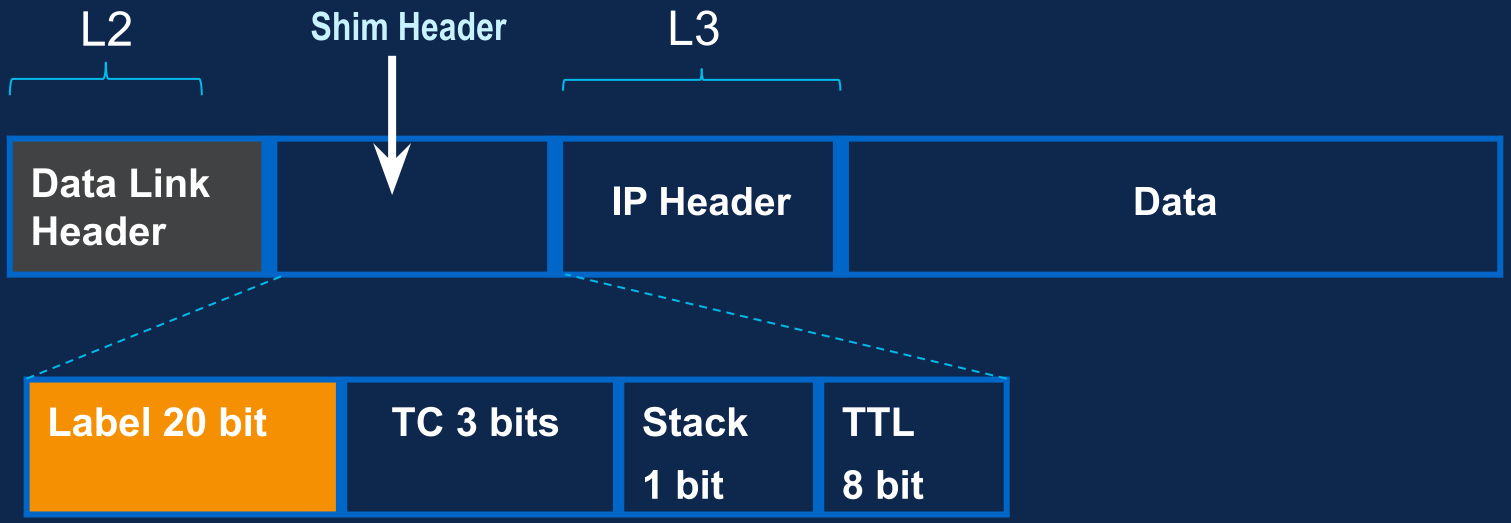
These were all similar technologies but were NOT interoperable.

- IETF worked to standardize the solutions and develop MPLS:-
  - First IETF RFC on MPLS published in Jan 2001
    - ❖ RFC 3031 specifies "MPLS Architecture" (https://tools.ietf.org/html/rfc3031)
    - ❖ RFC 3032 specify "Label Stack Encoding"(https://tools.ietf.org/html/rfc3032)

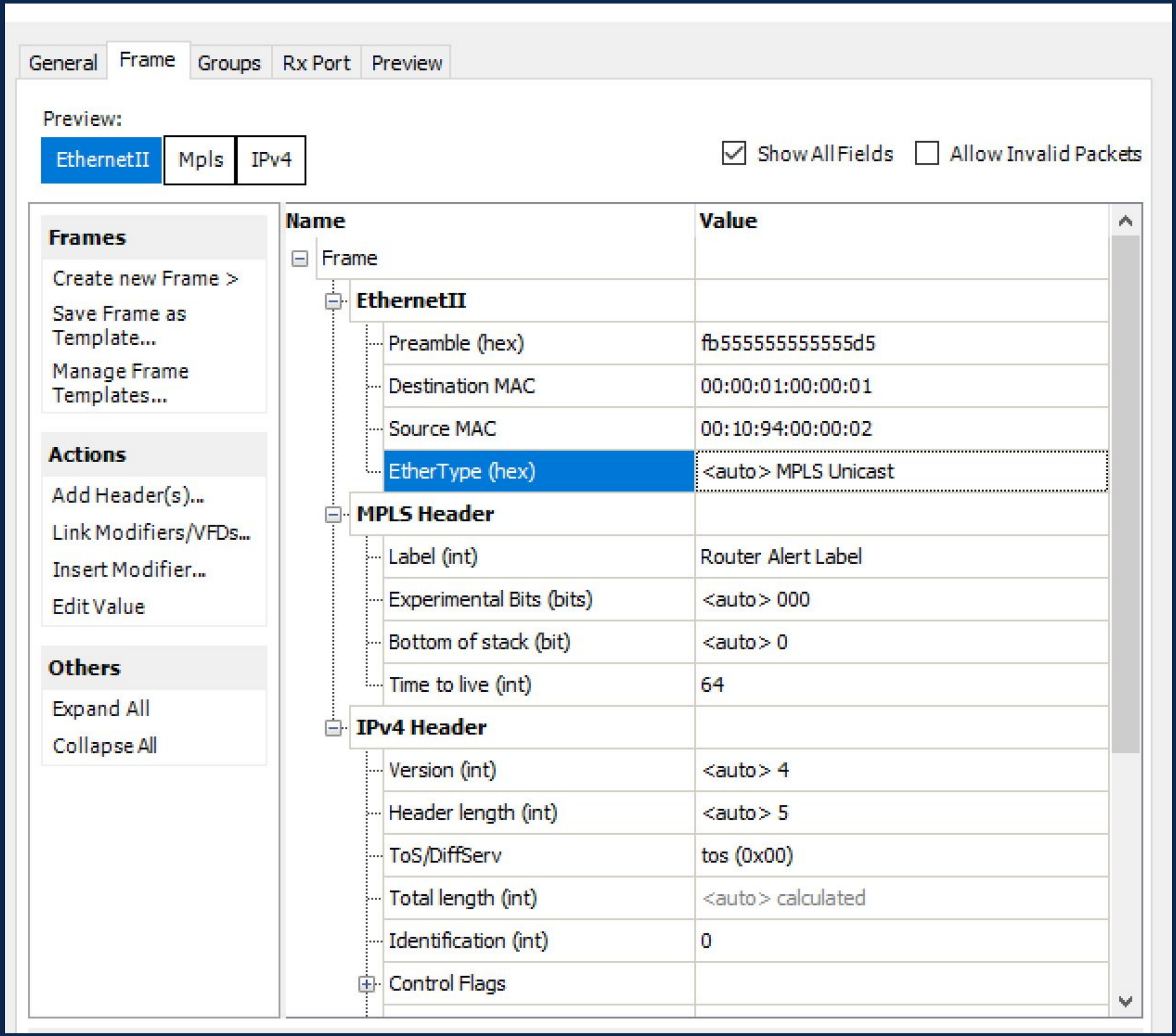
MPLS Data Plane

- It's a simple encapsulation at "layer 2.5"
- MPLS forwards packets by switching labels [push, pop] based on a forwarding database.
- Forwarding is based on simple look-up.

{incoming interface, incoming label} {outgoing interface, outgoing label}



MPLS FRAME


General
Frame
Groups
Rx Port
Preview
Preview:
EthernetII
Mpls
IPv4
Show All Fields
Allow Invalid Packets
Frames
Create new Frame >
Save Frame as Template...
Manage Frame Templates...
Actions
Add Header(s)...
Link Modifiers/VFDs...
Insert Modifier...
Edit Value
Others
Expand All
Collapse All
Name
Value
Frame
EthernetII
Preamble (hex)
fb555555555555d5
Destination MAC
00:00:01:00:00:01
Source MAC
00:10:94:00:00:02
EtherType (hex)
<auto> MPLS Unicast
MPLS Header
Label (int)
Router Alert Label
Experimental Bits (bits)
<auto> 000
Bottom of stack (bit)
<auto> 0
Time to live (int)
64
IPv4 Header
Version (int)
<auto> 4
Header length (int)
<auto> 5
ToS/DiffServ
tos (0x00)
Total length (int)
<auto> calculated
Identification (int)
0
Control Flags

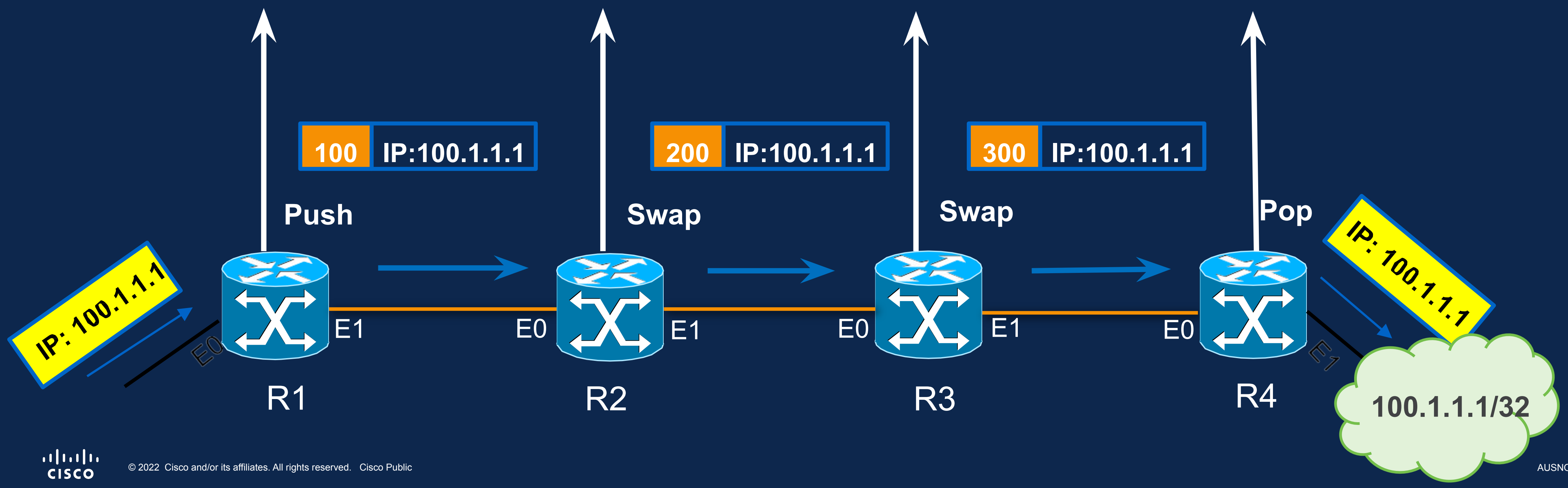
MPLS Forwarding Plane Operations

Prefix: 100.1.1.1/32	
Local Label	Null
Out Interface	E1
Out Label	100
Operation	Push

Prefix: 100.1.1.1/32	
Local Label	100
Out Interface	E1
Out Label	200
Operation	Swap

Prefix: 100.1.1.1/32	
Local Label	200
Out Interface	E1
Out Label	300
Operation	Swap

Prefix: 100.1.1.1/32	
Local Label	300
Out Interface	--
Out Label	--
Operation	POP



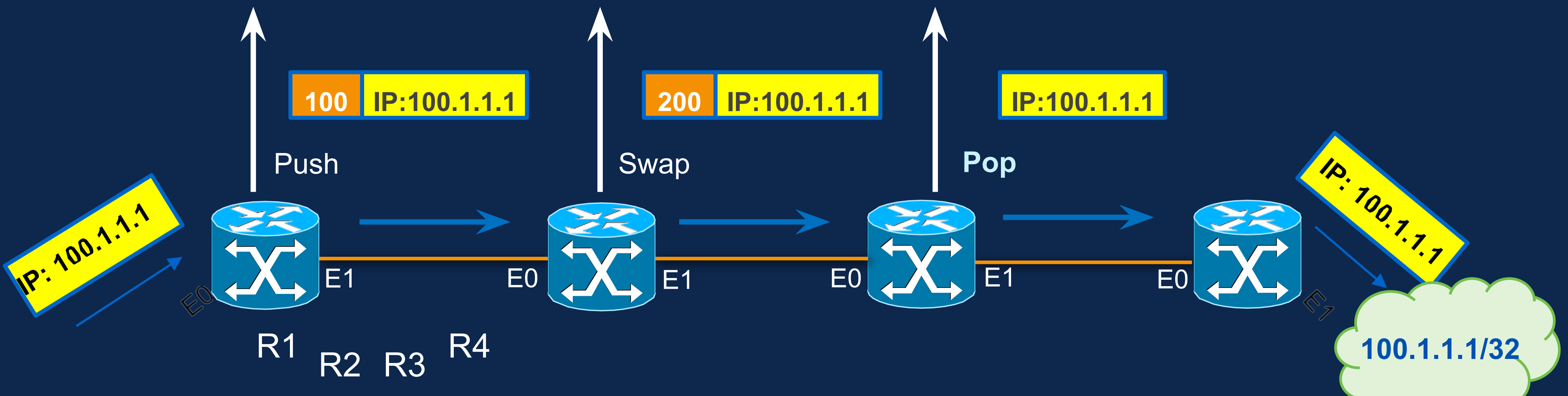
MPLS: Penultimate hop popping (PHP)

Prefix: 100.1.1.1/32	
Local Label	Null
Out Interface	E1
Out Label	100
Operation	Push

Prefix: 100.1.1.1/32	
Local Label	100
Out Interface	E1
Out Label	200
Operation	Swap

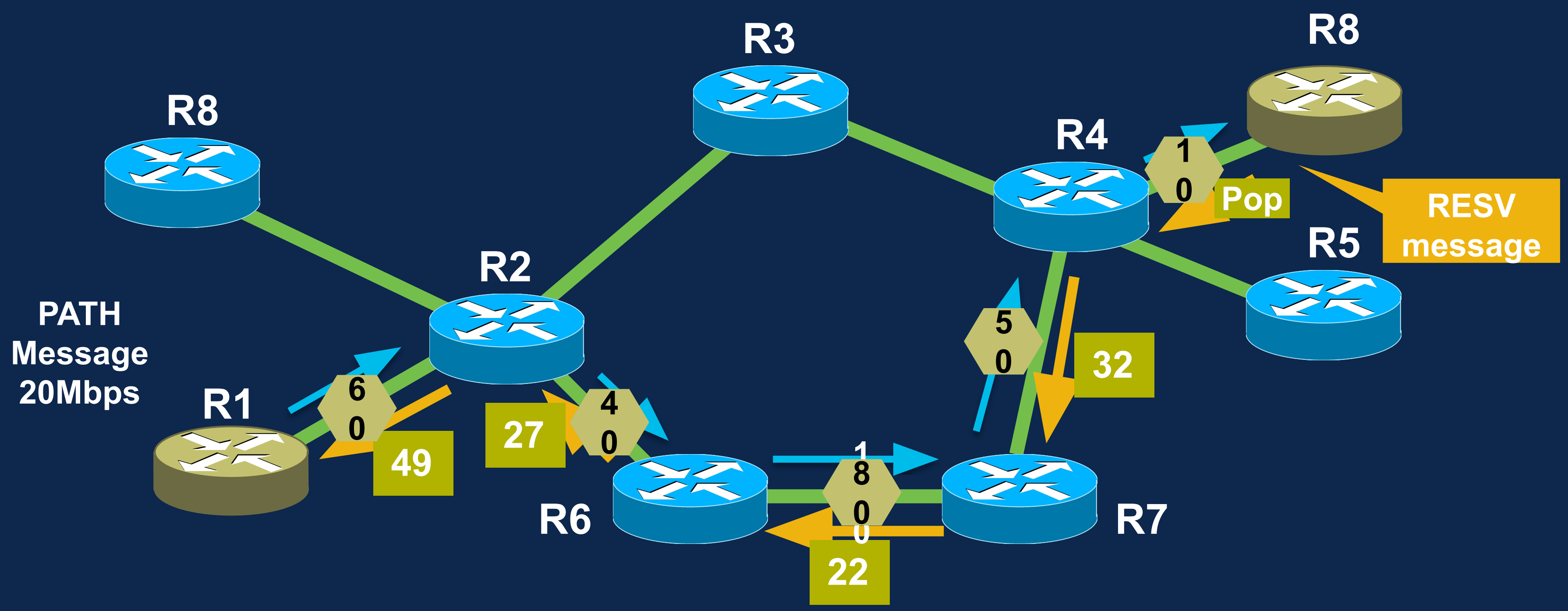
Prefix: 100.1.1.1/32	
Local Label	200
Out Interface	E1
Out Label	imp-null
Operation	Pop

Prefix: 100.1.1.1/32	
Local Label	imp-null
Out Interface	--
Out Label	--
Operation	--



* label with a value of 3 signifies an implicit NULL

RSVP – TE CONTROL PLANE



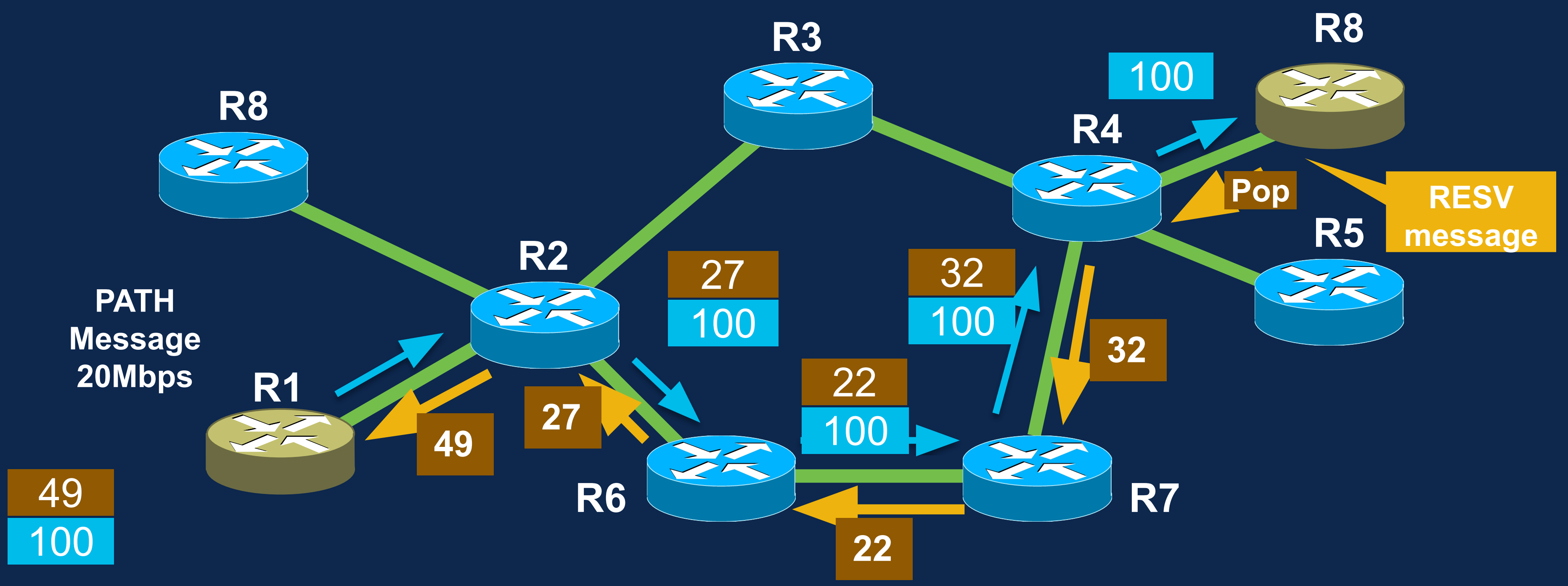
RSVP PATH: { R1 □ R2 □ R6 □ R7 □ R4 □ R9 }

RSVP RESV: Returns labels and reserves bandwidth on each link

Bandwidth available

Re ed label via RESV message

RSVP – TE DATA PLANE



Explicit Route Object { R1 □ R2 □ R6 □ R7 □ R4 □ R9 }

RSVP RESV: Returns labels and reserves

Service Label

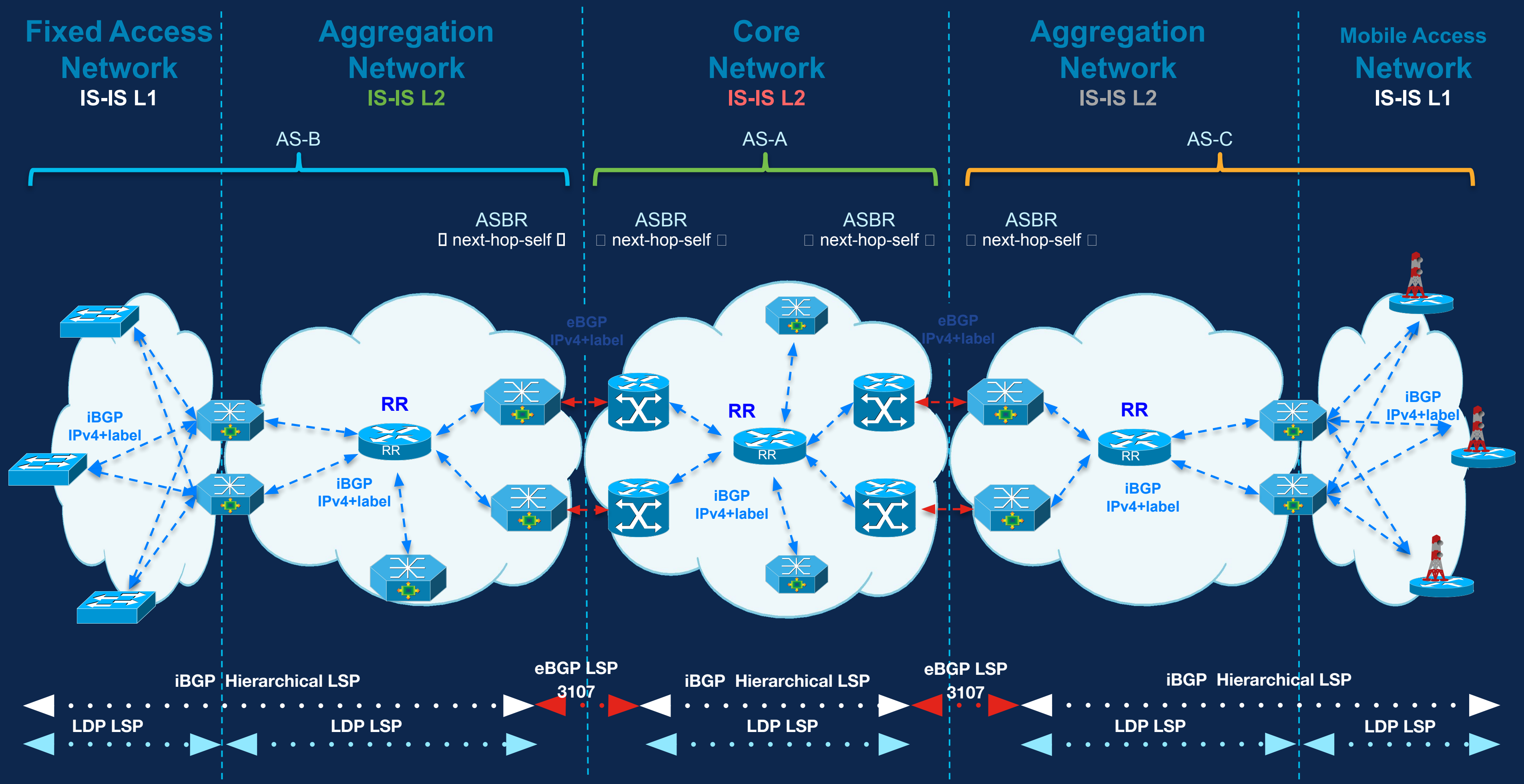
100

49

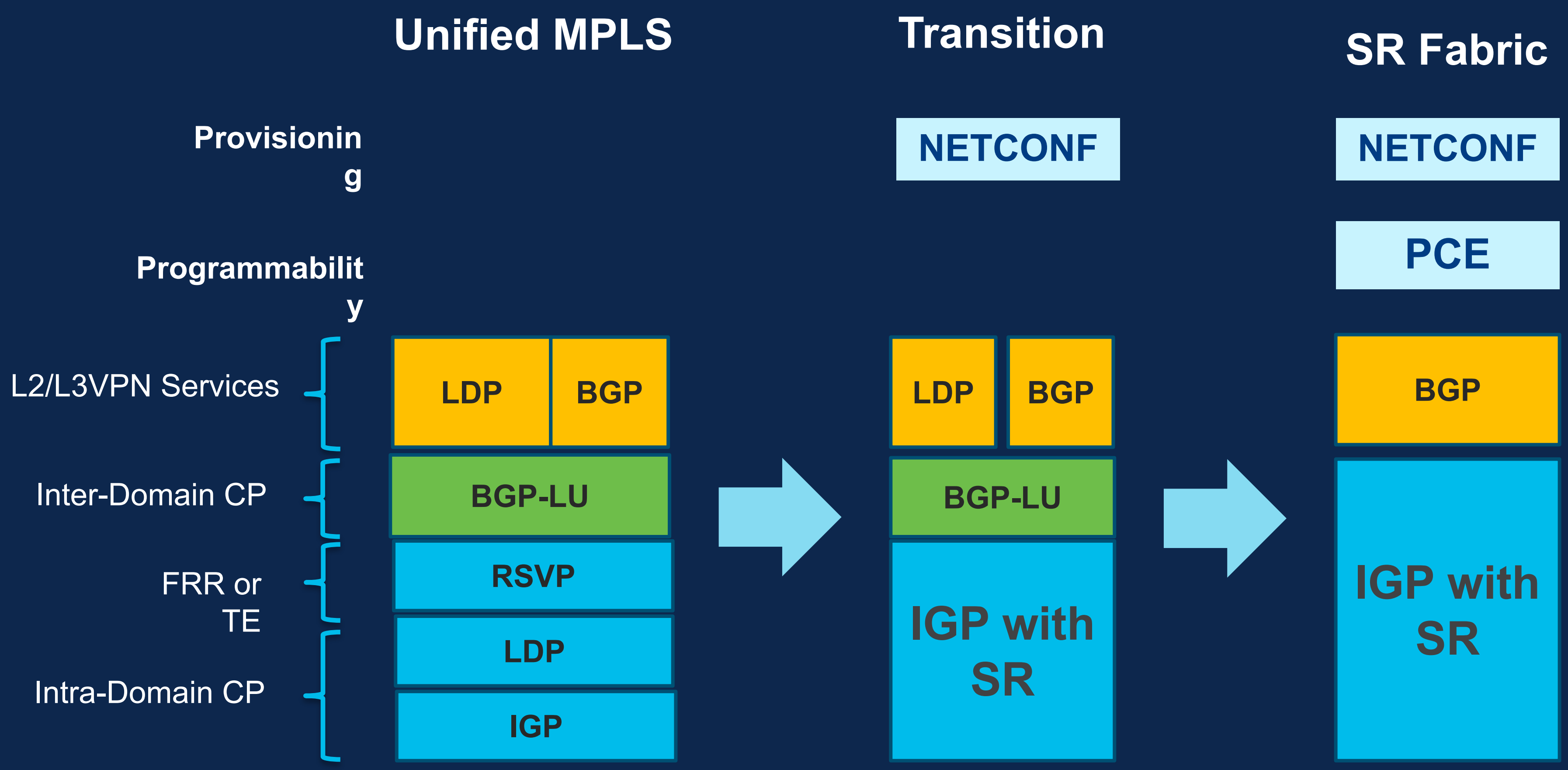
(y LDP, BGP etc.)

ed label via RESV message

and that got us here ..Unified MPLS Transport Model



SP Network - Simplification Journey



What is a “Segment” ?

“Segment”

“an instruction a node executes on the incoming packet”

Internet Engineering Task Force (IETF)
Request for Comments: **8402**
Category: Standards Track
ISSN: 2070-1721

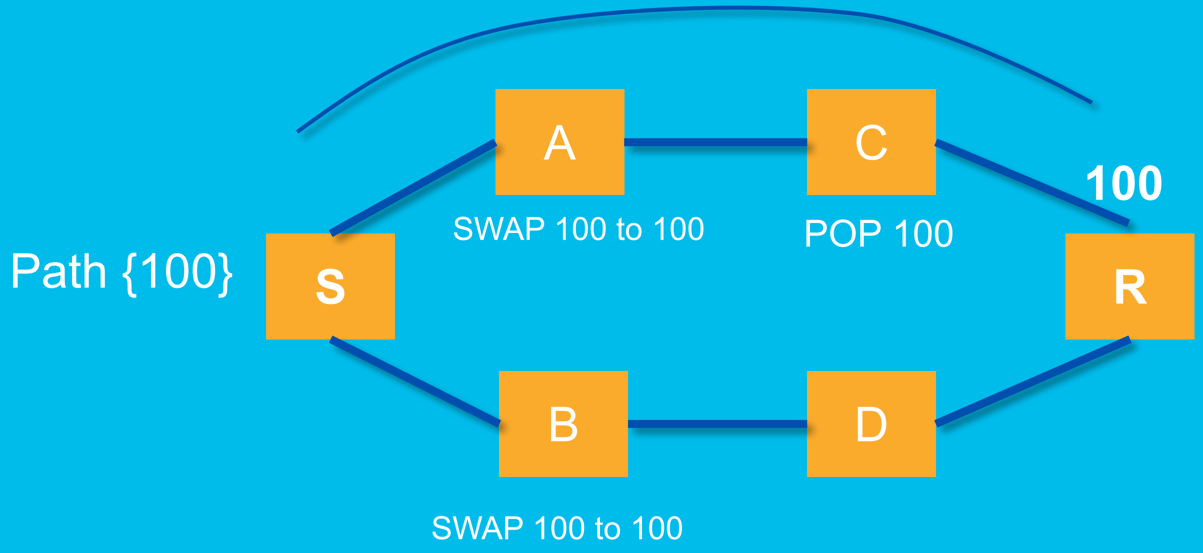
C. Filsfils, Ed.
S. Previdi, Ed.
L. Ginsberg
Cisco Systems, Inc.
B. Decraene
S. Litkowski
Orange
R. Shakir
Google, Inc.
July 2018

Segment Routing Architecture

Abstract

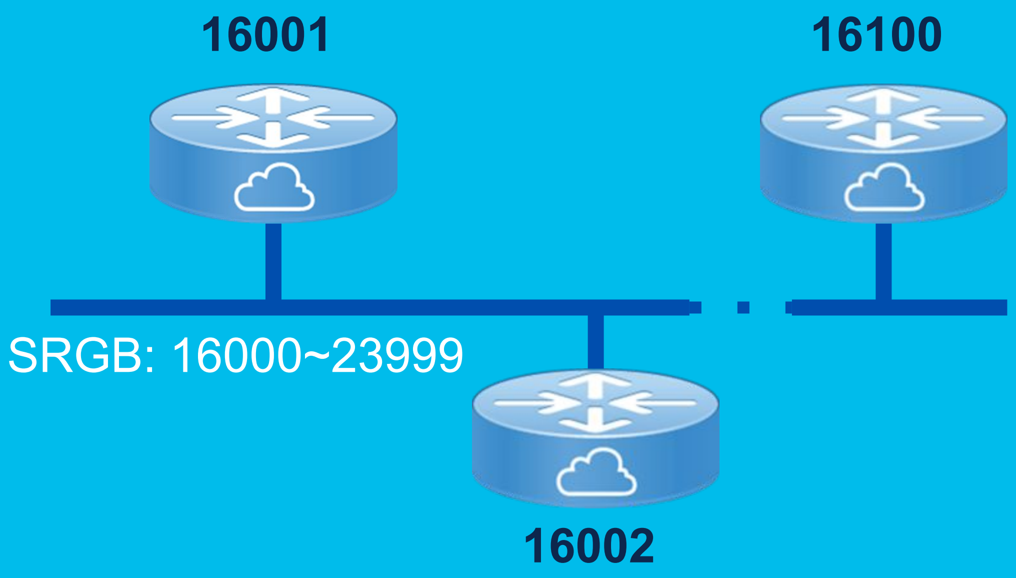
Segment Routing (SR) leverages the source routing paradigm. A node steers a packet through an ordered list of ***instructions***, called **"segments"**. A segment can represent any **instruction**, ***topological*** or ***service based***.

SRGB, IGP-Node Segment and IGP-Prefix Segment



IGP PREFIX SEGMENT

- Signaled by ISIS/OSPF
  - Minor extensions to existing link-state routing protocols
- Shortest-path to IGP prefix
  - Equal Cost Multipath (ECMP)-aware
- Global significance in SR domain
- **Label = SRGB + Index**
  - SRGB = Segment Routing Global Block
  - Default SRGB: 16,000 – 23,999
  - Advertised as index



Constructs of Segment Routing - SRGB and Prefix SID – Control Plane

```
RP/0/RP0/CPU0:PTT-5504-1#show isis database  verbose | i "SRGB"
Wed Aug 10 01:55:53.556 UTC
   Segment Routing: I:1 V:0, SRGB Base: 16000 Range: 8000
   Segment Routing: I:1 V:0, SRGB Base: 16000 Range: 8000
   Segment Routing: I:1 V:0, SRGB Base: 16000 Range: 8000
```

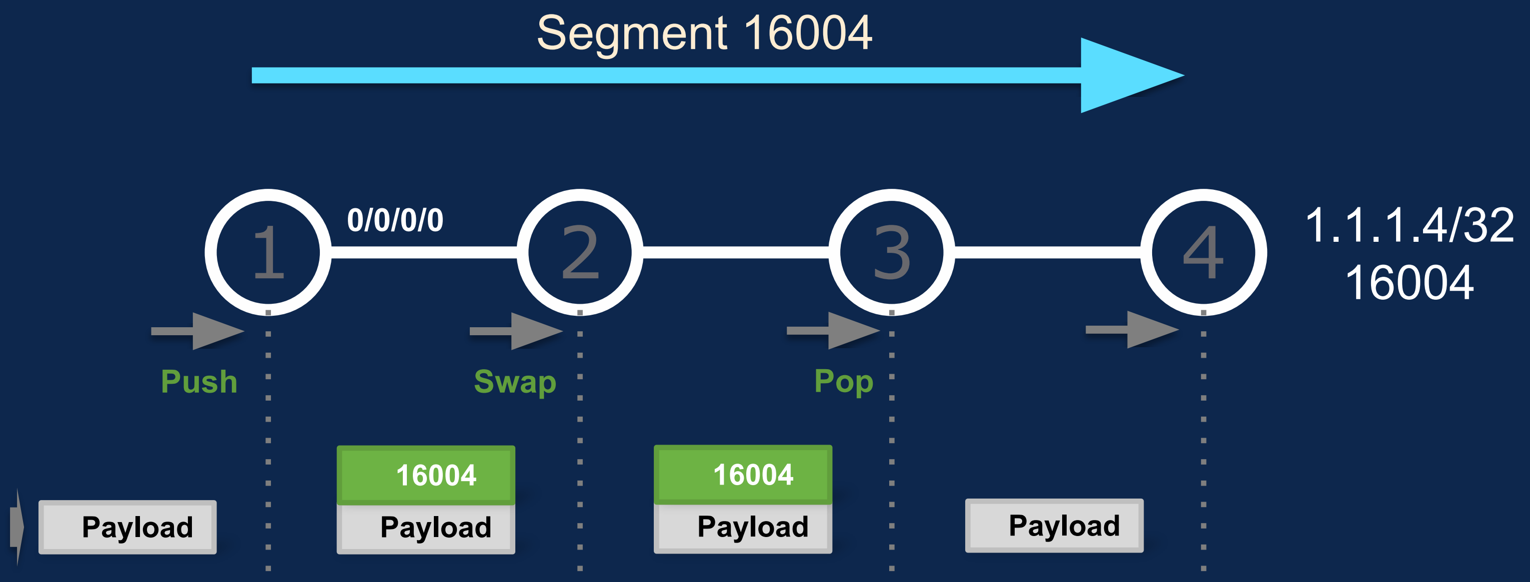
```
RP/0/RP0/CPU0:PTT-5504-1#show isis database  verbose | i "Prefix-SID Index:"
Wed Aug 10 01:58:28.622 UTC
   Prefix-SID Index: 1, Algorithm:0, R:0 N:1 P:0 E:0 V:0 L:0
   Prefix-SID Index: 801, Algorithm:128, R:0 N:1 P:0 E:0 V:0 L:0
   Prefix-SID Index: 901, Algorithm:129, R:0 N:1 P:0 E:0 V:0 L:0
   Prefix-SID Index: 2, Algorithm:0, R:0 N:1 P:0 E:0 V:0 L:0
   Prefix-SID Index: 802, Algorithm:128, R:0 N:1 P:0 E:0 V:0 L:0
   Prefix-SID Index: 902, Algorithm:129, R:0 N:1 P:0 E:0 V:0 L:0
   Prefix-SID Index: 3, Algorithm:0, R:0 N:1 P:0 E:0 V:0 L:0
   Prefix-SID Index: 803, Algorithm:128, R:0 N:1 P:0 E:0 V:0 L:0
   Prefix-SID Index: 903, Algorithm:129, R:0 N:1 P:0 E:0 V:0 L:0
```

```
router isis core

 flex-algo 128
 !
 flex-algo 129
 !
 address-family ipv4 unicast
  metric-style wide
  segment-routing mpls

 interface Loopback0
  passive
  address-family ipv4 unicast
   prefix-sid absolute 16003
   prefix-sid algorithm 128 absolute 16803
   prefix-sid algorithm 129 absolute 16903
```

SR – DATA PLANE OPERATION




```
RP/0/0/CPU0:Node1#show cef 1.1.1.4/32
1.1.1.4/32, version 277, internal 0x4004001 0x0 (ptr 0xacce39a4) [1], 0x0 (0xaccde760), 0x450 (0xacd8b8)
 local adjacency 10.0.0.2
 Prefix Len 32, traffic index 0, precedence n/a, priority 1
   via 99.1.2.2, GigabitEthernet0/0/0/0, 5 dependencies, weight 0, class 0 [flags 0x0]
    path-idx 0 NHID 0x0 [0xacbb3bf0 0x0]
    next hop 99.1.2.2
    local adjacency
     local label 16004      labels imposed {16004}
```

IGP-Adjacency SID

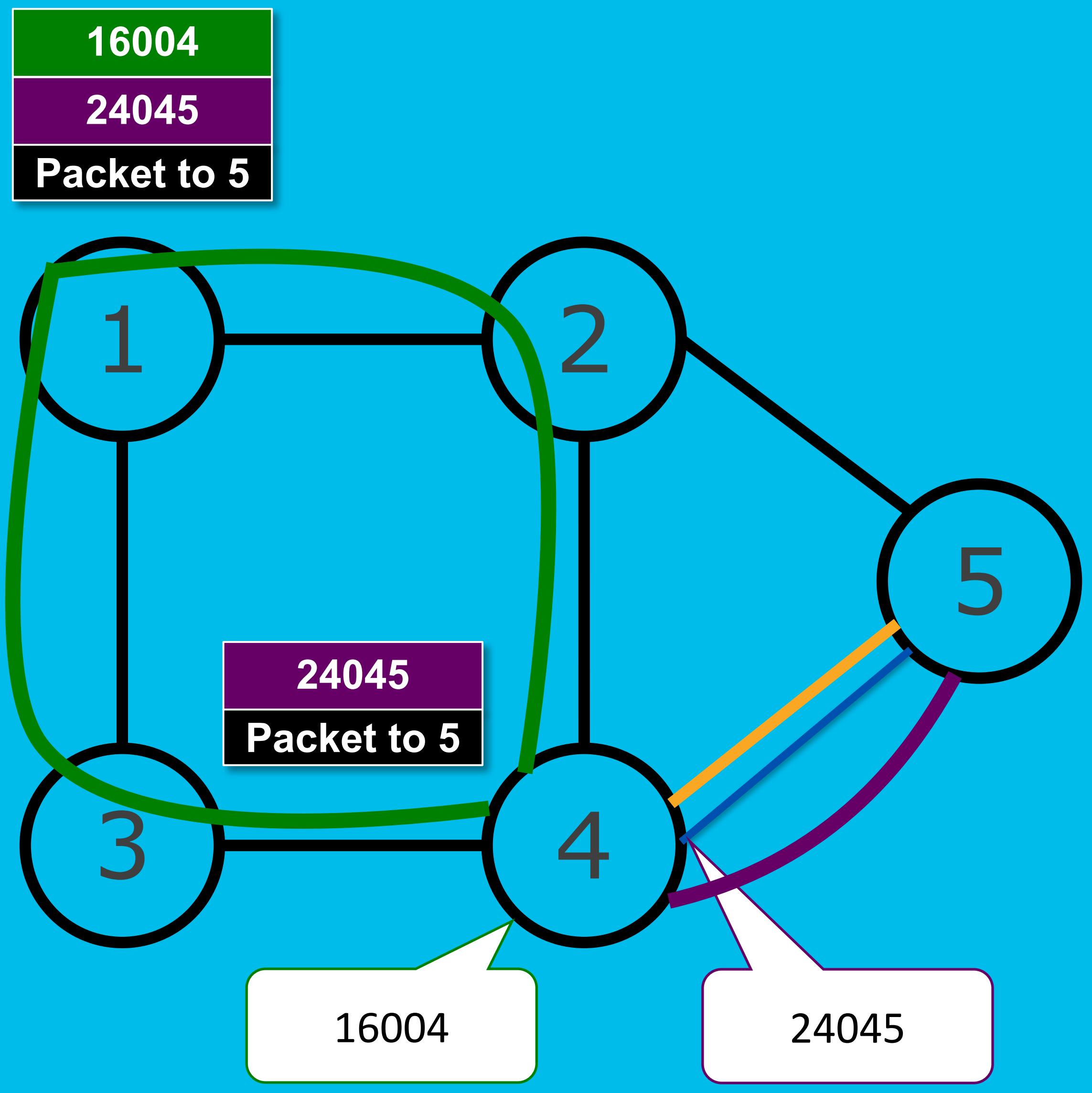
Steer traffic on any path through the network

Path is specified by list of segments in packet header, a stack of labels

No path is signalled

No per-flow state is created

Single protocol: IS-IS or OSPF



ANYCAST SID

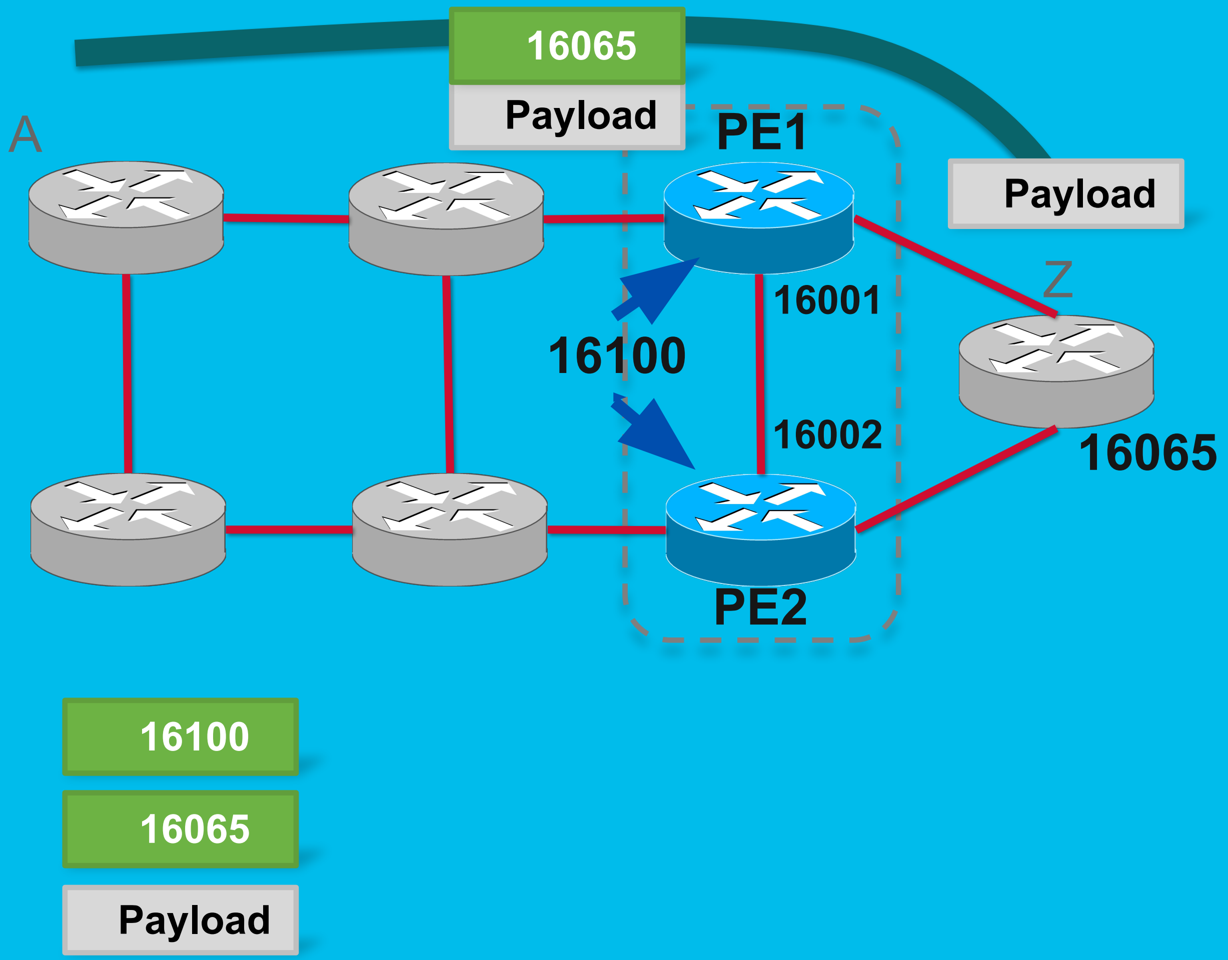
Anycast prefixes: same prefix advertised by multiple nodes

Anycast prefix-SID: Same prefix-SID for the same prefix!

Traffic is forwarded to one of the Anycast prefix-SID originators, **based on best IGP path**

If primary node fails, traffic is auto re-routed to the other node

Note: *nodes advertising the same Anycast prefix-SID must have the same SRGB*



ANYCAST SID

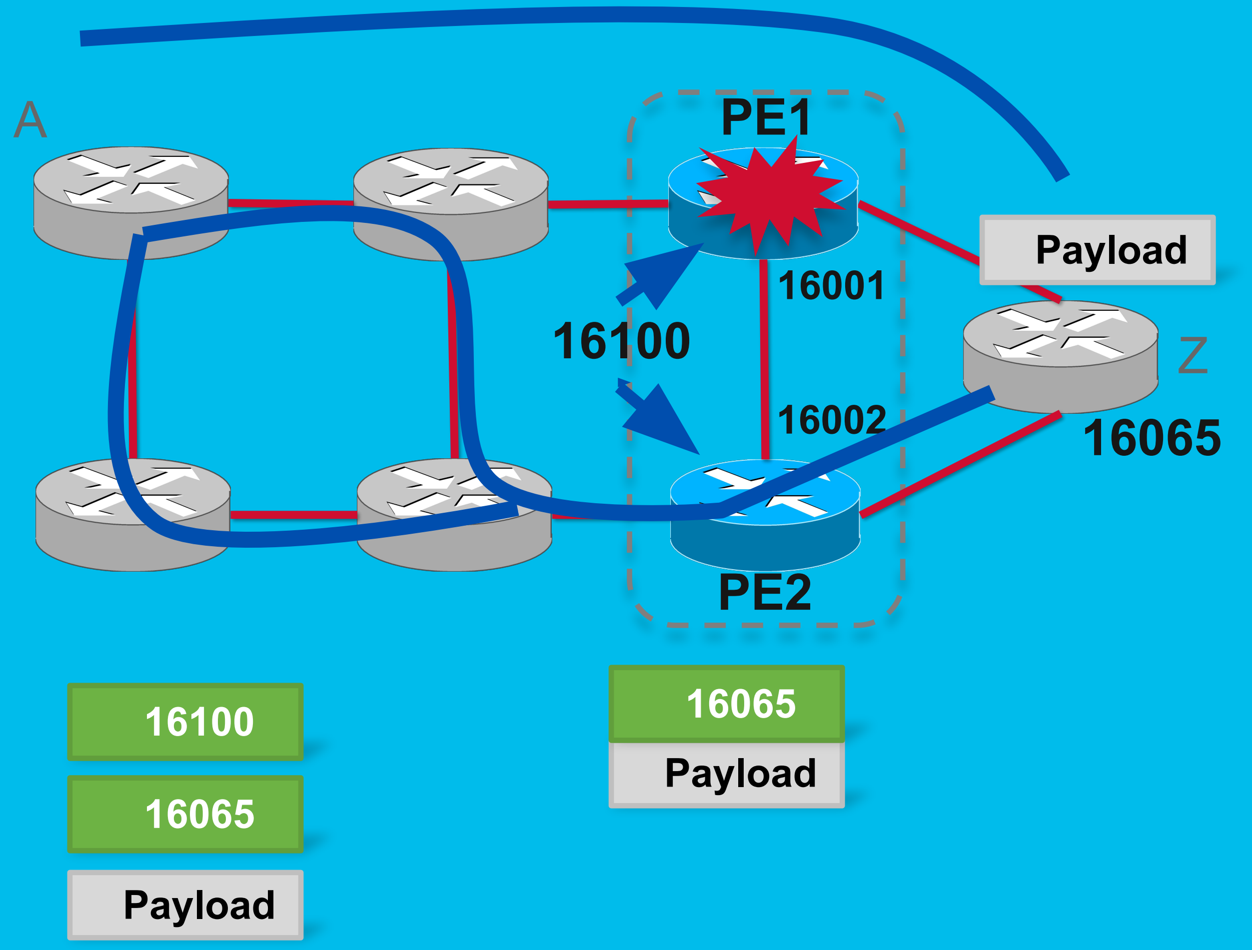
Anycast prefixes: same prefix advertised by multiple nodes

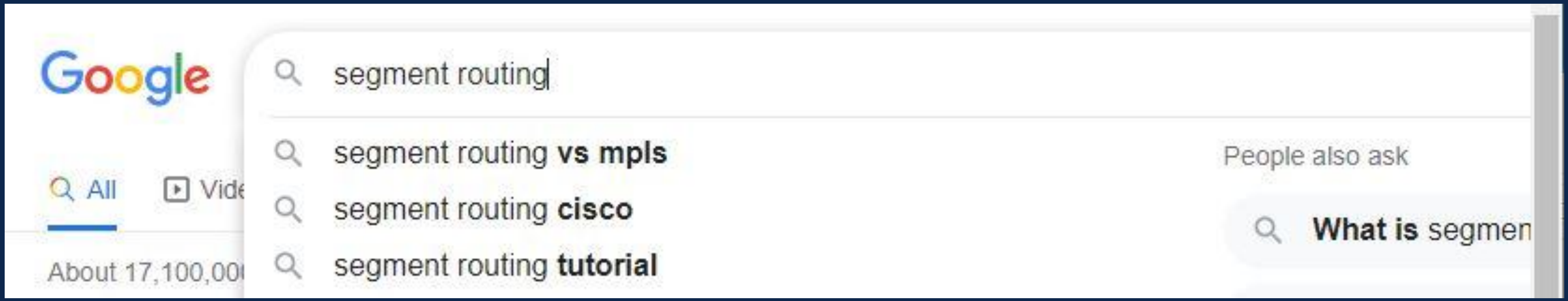
Anycast prefix-SID: prefix-SID associated with anycast prefix - Same prefix-SID for the same prefix!

Traffic is forwarded to one of the Anycast prefix-SID originators **based on best IGP path**

If primary node fails, traffic is auto re-routed to the other node

Note*: nodes advertising the same Anycast prefix-SID must have the same SRGB*





Segment Routing is not a mpls replacement

VPNv4

VPNv6

VPWS

VPLS

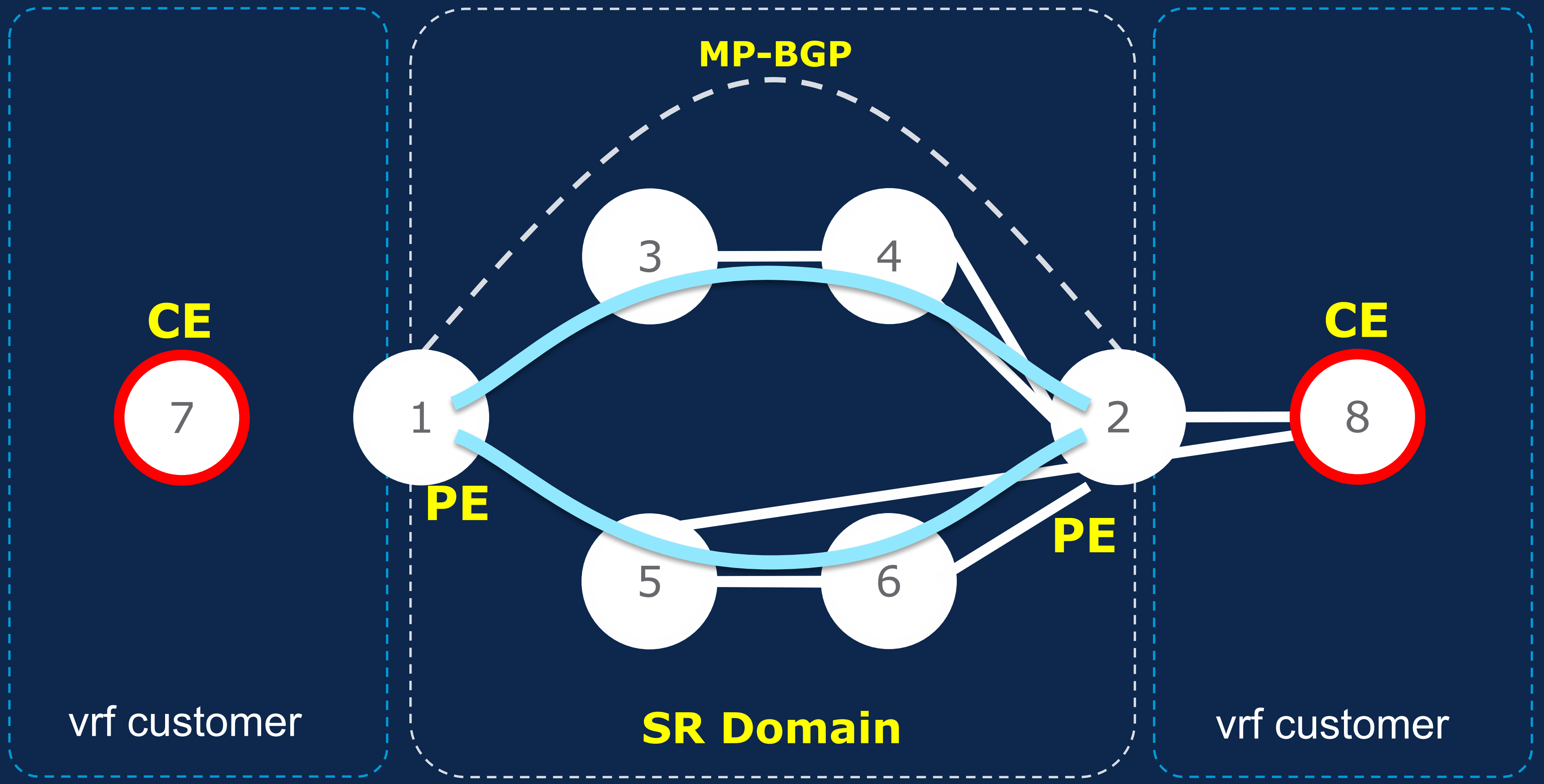
EVPN

Everything Used to Work with LDP

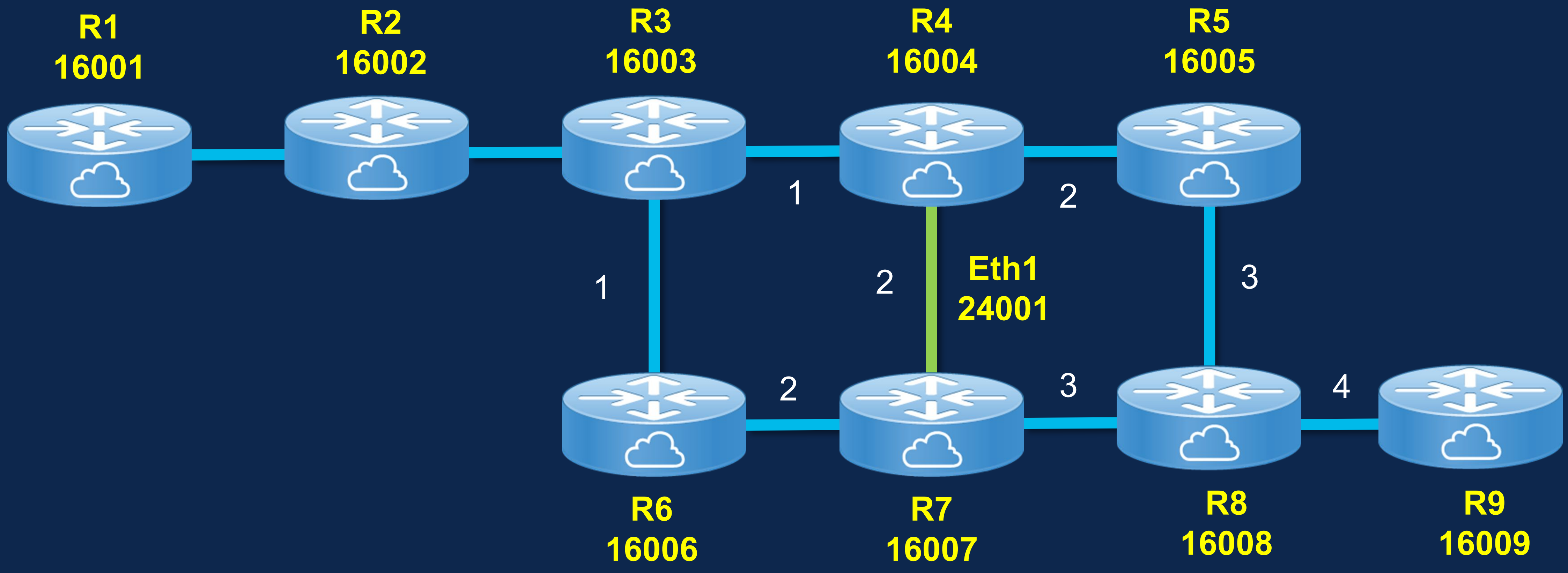
The overlay services will work as usual, including Inter-AS Options 1,2,3

OVERLAY SERVICE UNDERPINNED BY SR CORE

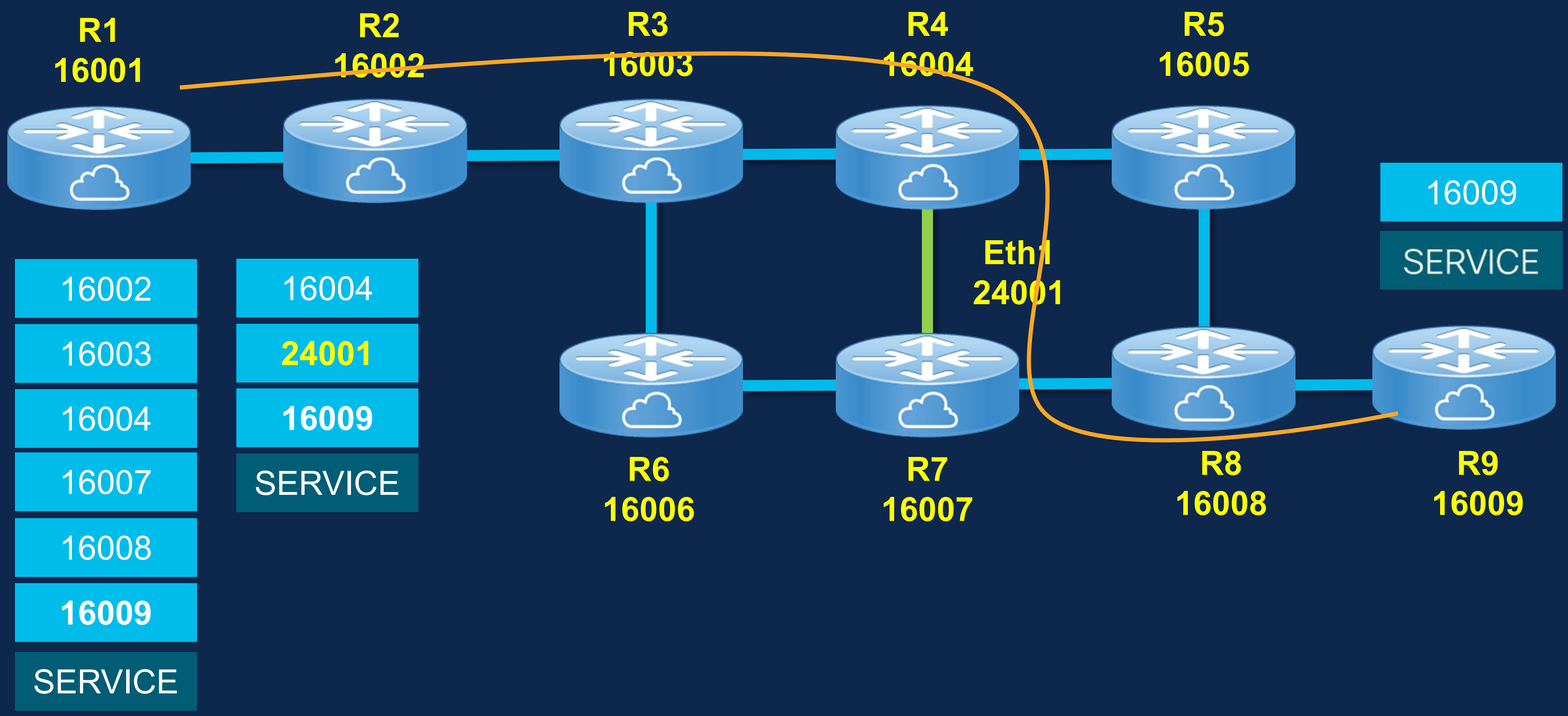
- MPLS services ride on prefix segments
- Simple, one less protocol to operate (LDP)



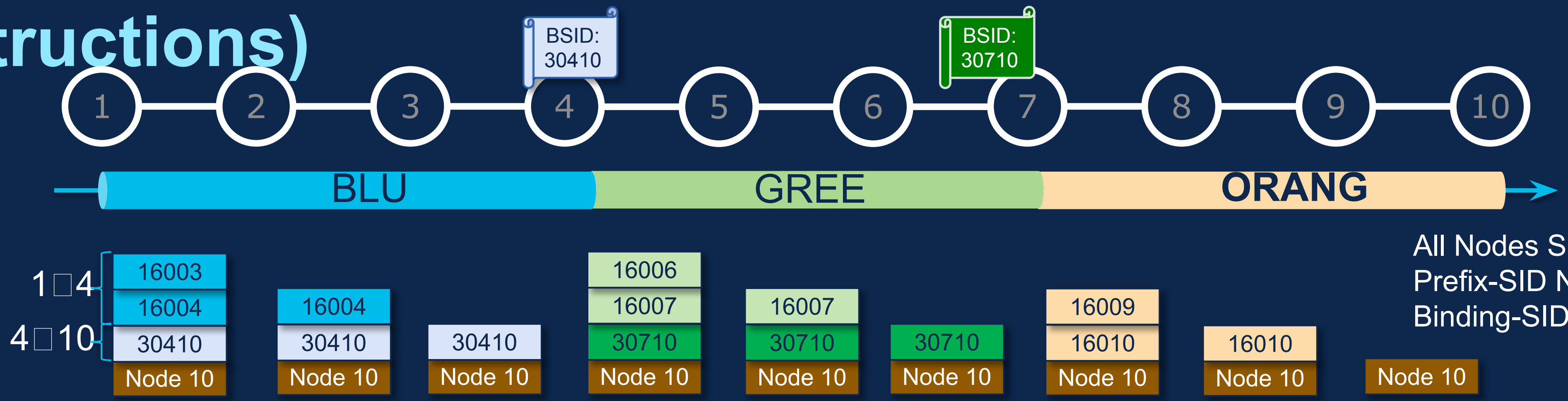
LABEL DEPTH ISSUES



LABEL DEPTH ISSUES



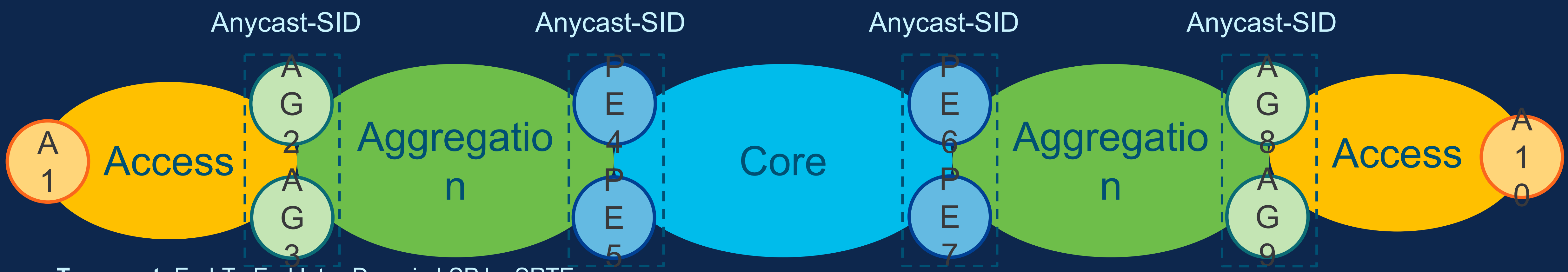
Binding-SID, Stitching Segments (Topological Instructions)



All Nodes SRGB [16,000-23,999]
Prefix-SID NodeX: 1600X
Binding-SID X□Y: 300XY

- Assume Node1 can’t push 8 labels to go to Node10
  - “compress” label stack by stitching SRTE Policies:
    - Node1 pushes:
      - 2 labels to go to Node4
      - Binding-SID (30410) to go to Node10
    - Node4 pops Binding-SID and pushes:
      - 2 labels to go to Node7
      - Binding-SID (30710) to go to Node10
    - Node7 pops Binding-SID and pushes 2 labels to go to Node10

Inter-Domain LSP - Data Plane



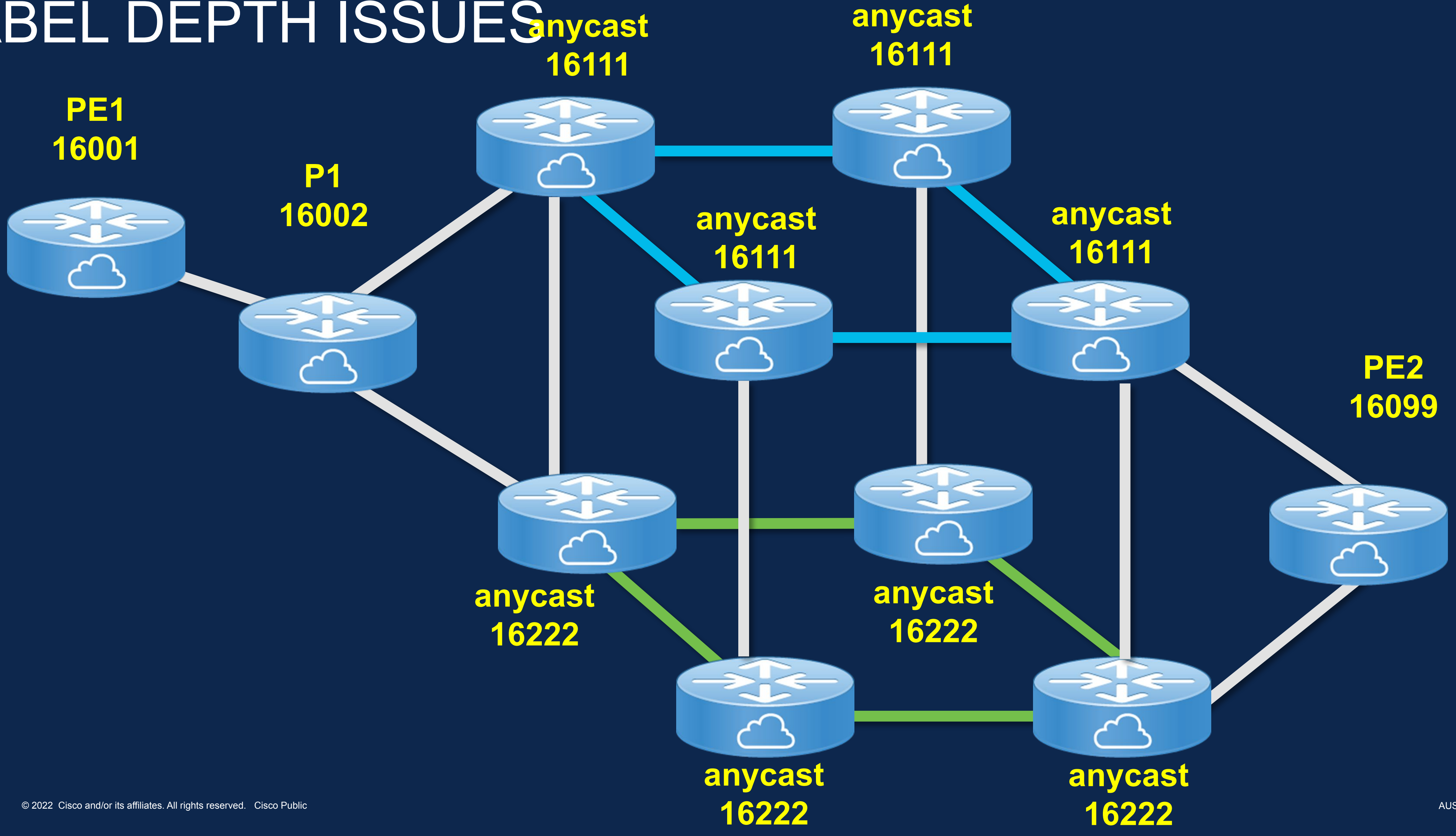
Transport: End-To-End Inter-Domain LSP by SRTE

TI-LFA	TI-LFA	TI-LFA	TI-LFA	TI-LFA
TI-LFA	TI-LFA	TI-LFA	TI-LFA	TI-LFA
AG2/3	PE4/5	PE6/7	AG8/9	A10
PE4/5	PE6/7	AG8/9	A10	
PE6/7	AG8/9	A10		
AG8/9	A10			
A10				

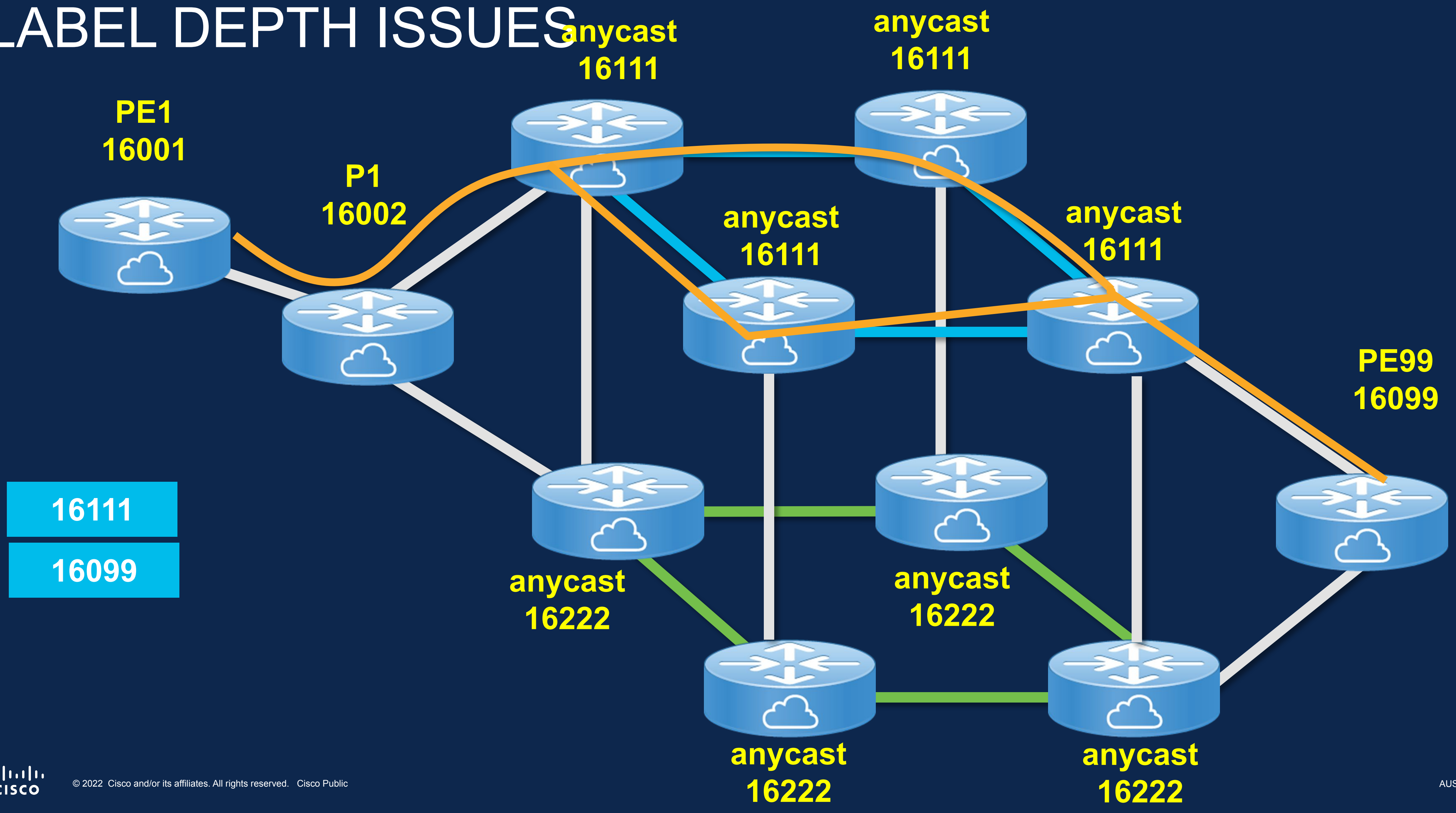
Label Stack must be optimized!
Service label(s) need to be considered!

CHALLENGES

LABEL DEPTH ISSUES



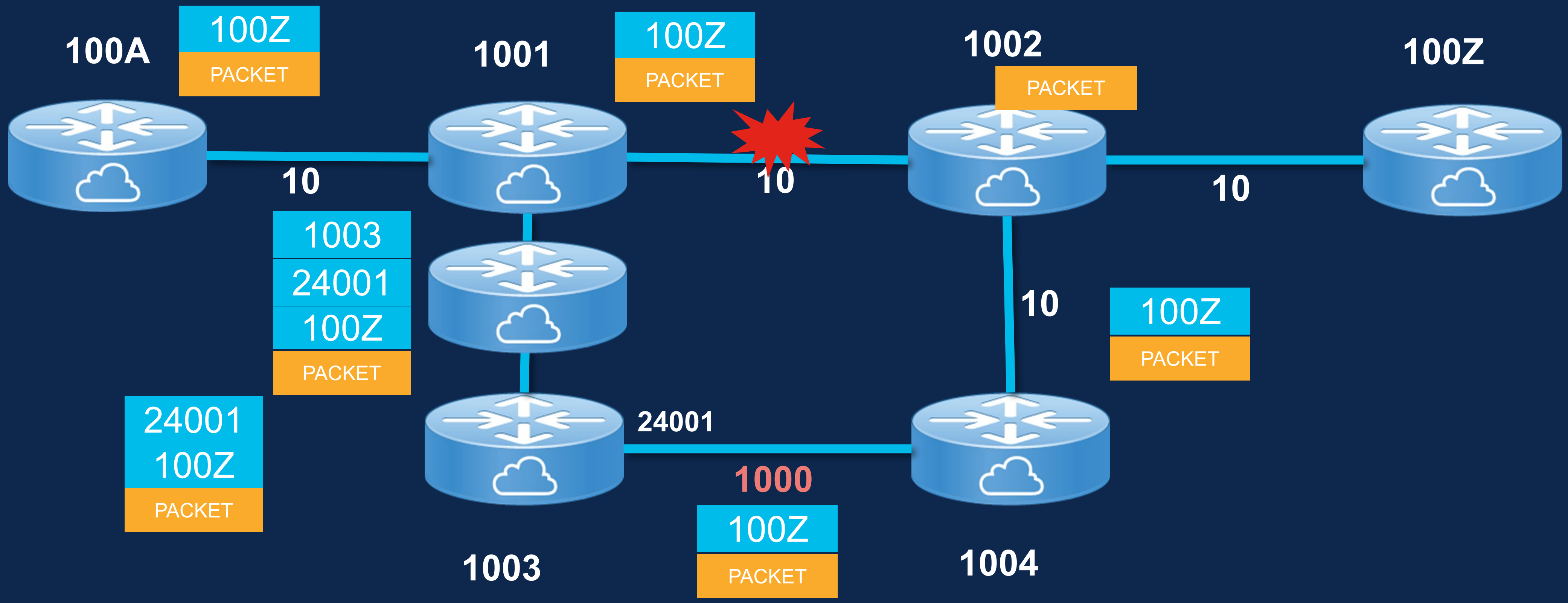
LABEL DEPTH ISSUES



What is TI-LFA

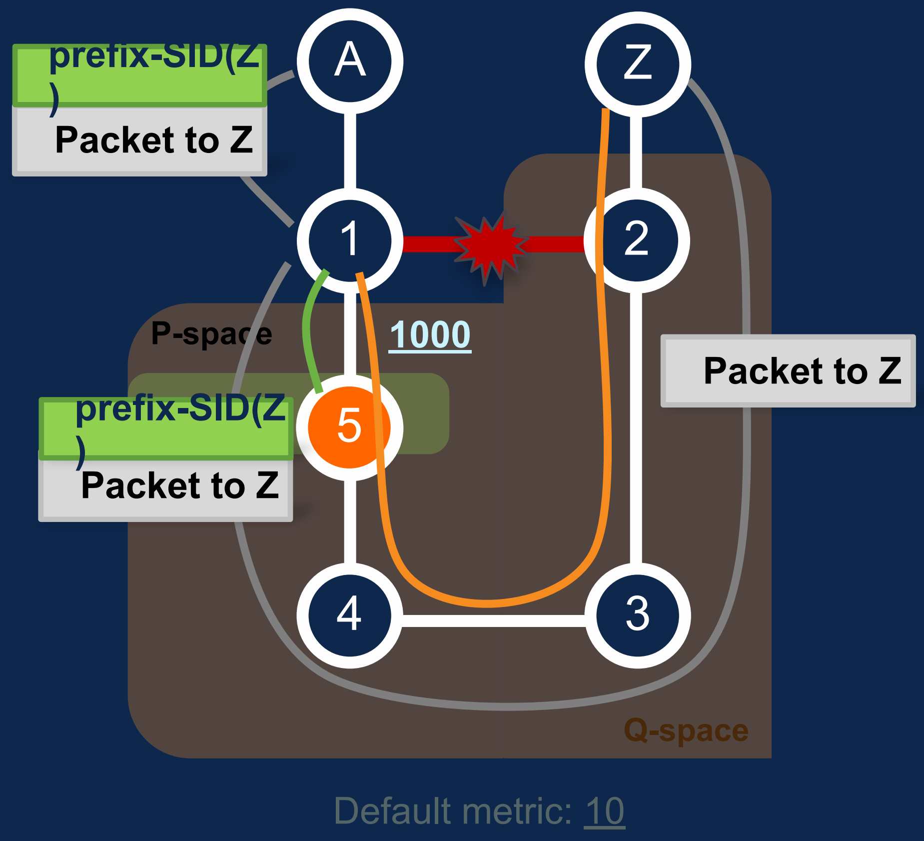
- ✔ **Extremely simple to configure**
- ✔ **Works on any topology (Topology Independent)**
- ✔ **Under 50ms restoration**
- ✔ **No Signaling**
- ✔ **Link or Node Protection**
- ✔ **Using Segments to force traffic over backup path**

TI-LFA Operation



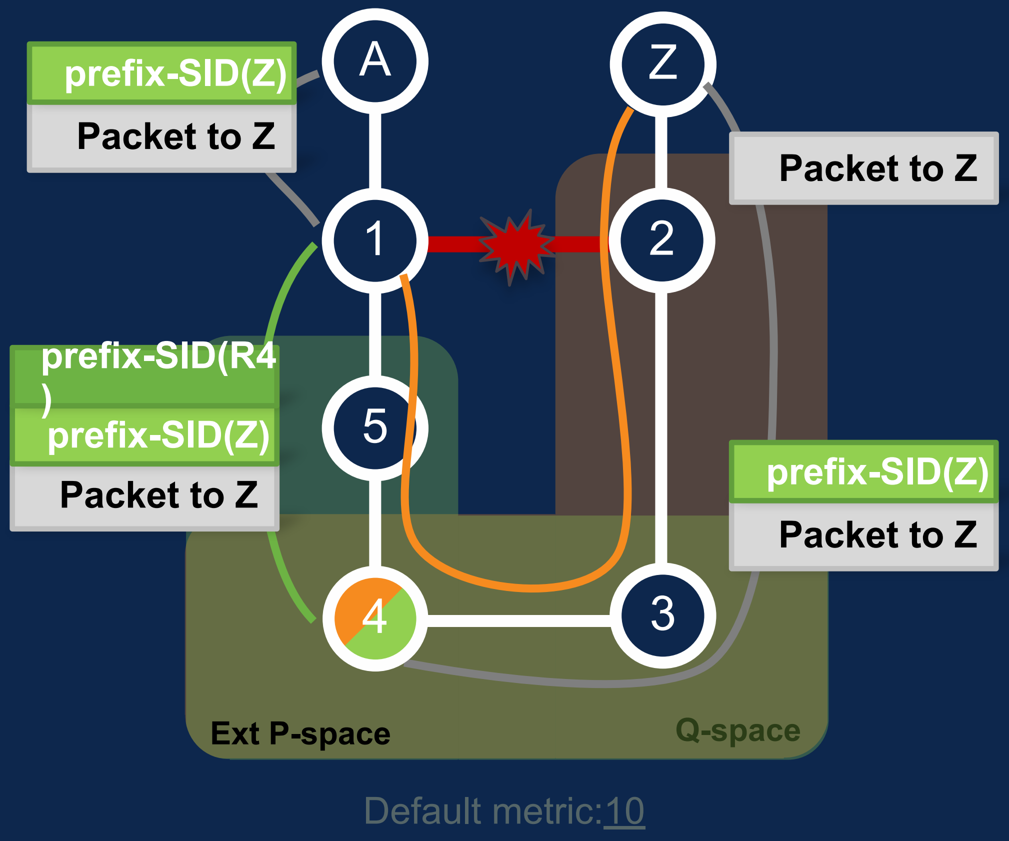
TI-LFA – zero-segment example

- To steer packets on TI-LFA backup path:
  - “forward packet to R5 without any additional segment”



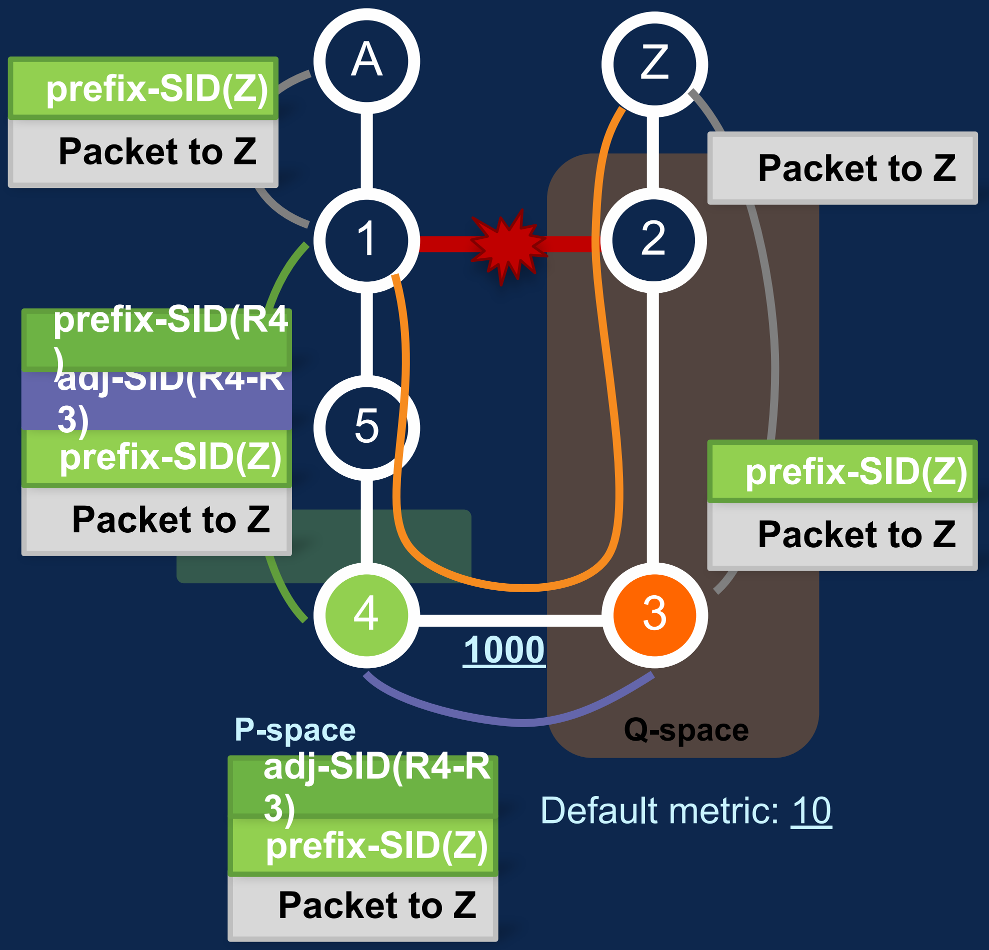
TI-LFA – single-segment example

- To steer packets on TI-LFA backup path:
  - “forward packet on interface to R5
  - push segment {prefix-SID(R4)}”



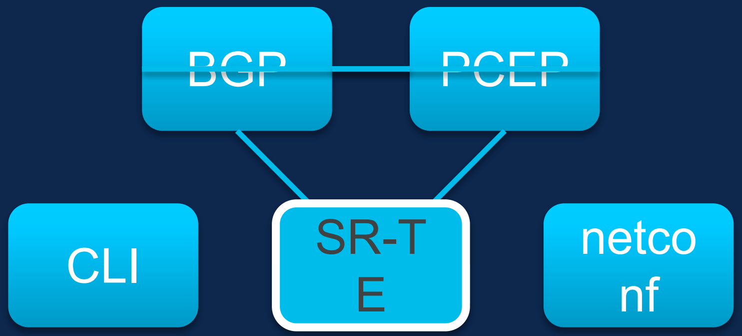
TI-LFA – double-segment example

- To steer packets on TI-LFA backup path:
  - forward packet on interface to R5
  - push segments {prefix-SID(R4) and adj-SID(R4-R3)}"



Traffic Engineering with Segment Routing

- Automated On-Demand Policy
  - No core state: state in the packet header
  - Automated steering: BGP prefix coloring
  - No tunnel interface: on-demand policy instantiation
- CLI or Controller Based Policy
  - Support constraint-based routing
  - CLI is just one way, more programmable ways
  - Support controller based BGP sr-policy instantiation
- Centralized Multi-Domain Policy
  - SR Path Compute Element (SR-PCE) for path compute
  - Binding-SID (BSID) for traffic steering and scale
  - Supports centralized policy via PCEP provision
- An SR Policy is identified through the following tuple:
  - The head-end where the policy is instantiated/implemented
  - The endpoint (i.e.: the destination of the policy)
  - The colour (an arbitrary numerical value)



No Signaling Protocol

Protection is Ti-LFA

ECMP by native

SR Policy CLI Configuration

```
segment-routing
 global-block 16000 23999
 traffic-eng
  logging
   policy status
  !
  segment-list explicit-to-ABR-1
   index 5 address ipv4 10.1.3.3
   index 10 mpls label 16005
  !
  policy to-ABR1
   binding-sid mpls 1000
   color 1000 end-point ipv4 10.0.0.5
   candidate-paths
    preference 100
     dynamic
      metric
       type igp
      !
     !
    !
    preference 200
     explicit segment-list explicit-to-ABR-1
```

SRTE

Explicit-path definition

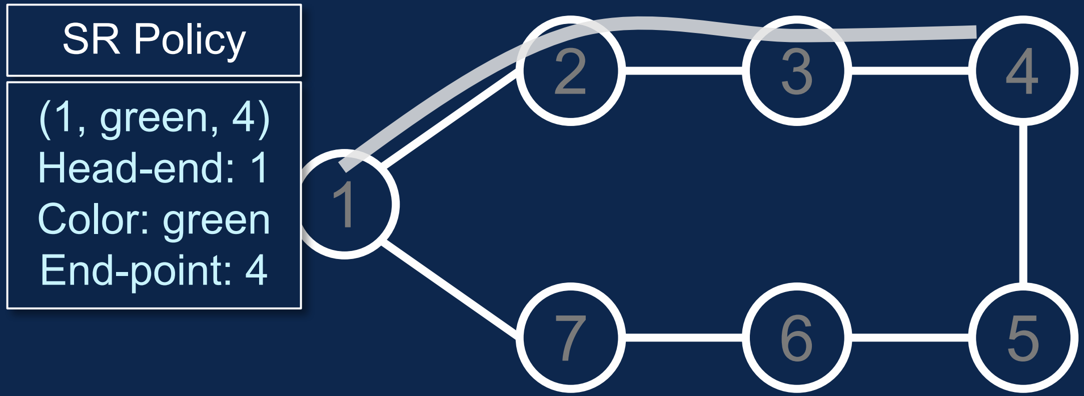
SR Policy

Dynamic path

Explicit path

candidate-paths with higher preference wins, if the path is valid

Examples of SR – POLICY usage



Configuring Binding SID

```
segment-routing
 traffic-eng
  policy AUSNOG-BSID
   binding-sid mpls 1000
   color 1000 end-point ipv4 1.1.1.4
   candidate-paths
    !
```

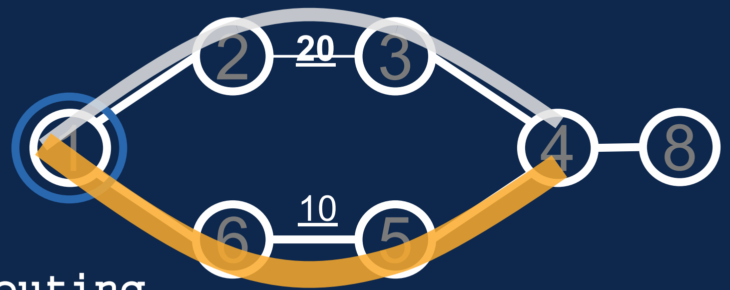
Static Route to an SR-POLICY

```
router static
 address-family ipv4 unicast
  1.1.1.4/32 sr-policy AUSNOG
 !
!
```

Per-Flow steering

```
policy per-flow-r4
 color 103 end-point ipv4 1.1.1.4
 candidate-paths
  preference 100
   per-flow
    forward-class 0 color 101
   !
```

SR Ping – Referencing a Policy



R1#ping sr-mpls policy name <POLICY1>
[lsp-end-point <1.1.1.8>]

Policy is referenced by its name.

User may optionally overwrite FEC value used in Ping.

R1#ping sr-mpls policy
color <20> end-point ipv4 <1.1.1.4>
[lsp-end-point <1.1.1.8>]

Policy is referenced by its color and end-point

R1#ping sr-mpls policy binding-sid 1000
[lsp-end-point <1.1.1.8>]

Policy is referenced by the binding SID.

The policy may be instantiated by configuration, netconf, PCEP or BGP-TE.

On Node1:

```
segment-routing
 traffic-eng
  policy POLICY1
   color 20 end-point ipv4 1.1.1.4
   binding-sid mpls 1000
   candidate-paths
    preference 200
     explicit segment-list SIDLIST1
      weight 1
     dynamic pce
      weight 4
    preference 300
      explicit segment-list SIDLIST2
      dynamic mpls
  segment-list name SIDLIST1
   index 10 mpls label 16002
   index 20 mpls label 24123
   index 30 mpls label 16004
  !
 segment-list name SIDLIST2
   index 10 address ipv4 1.1.1.4
```

Path Computation for Inter-Area

What is the solution?

- Centralized computation model using SR-PCE (Segment-Routing Path Computation Element)
- SR-PCE will compute & download path to Head-end or PCC (Path Computation Client)
- PCEP (Path Computation Element Protocol) is used to/from PCE and PCC or between PCEs

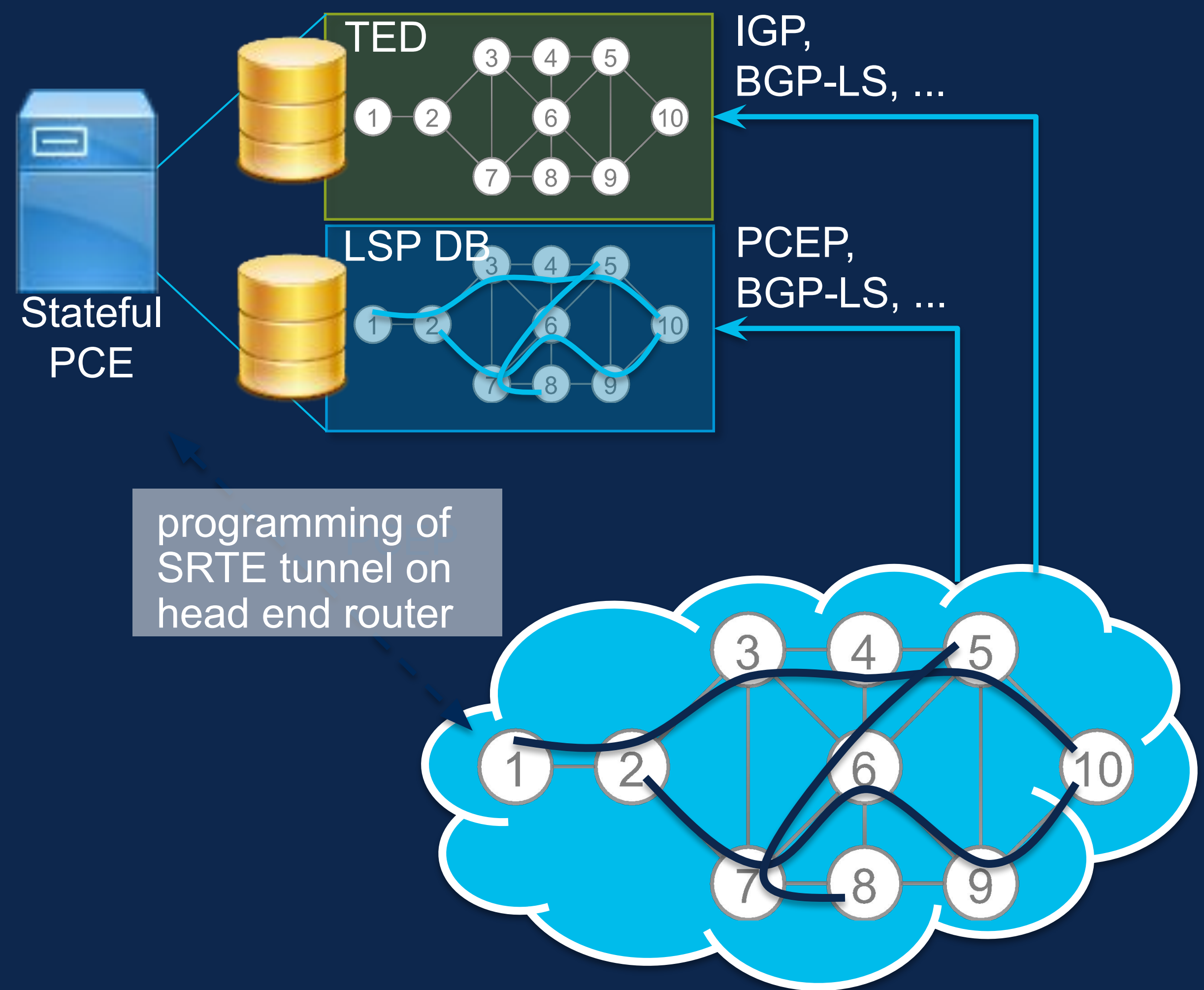
Path Computation Element

TED = IGP Topology + TE link attributes + SIDs + SRGB

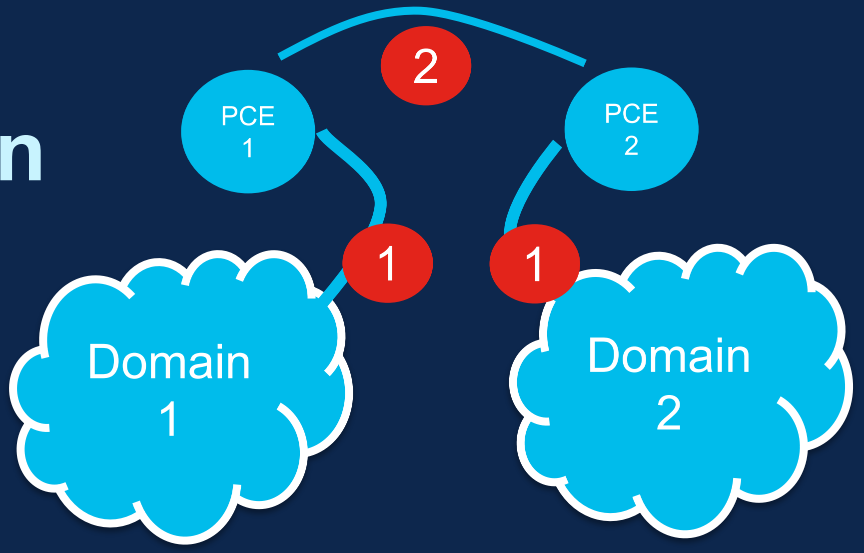
SRTE Policy path

TED = Traffic Engineering Database

- Path computation
- Large, multi-domain and multi-layer networks
- Path computation element (**PCE**)
  - ❖ Computes network paths (topology, paths, etc.)
  - ❖ Stores TE topology database (synchronized with network)
  - ❖ May initiate path creation
  - ❖ Stateful - stores path database included resources used (synchronized with network)
- Path computation client (**PCC**)
  - ❖ May send path computation requests to PCE
  - ❖ May send path state updates to PCE
- Used between head-end router (PCC) and PCE to:
  - ❖ Request/receive path from PCE subject to constraints
  - ❖ State synchronization between PCE and router
  - ❖ Hybrid CSPF



At a glance: SR-PCE Configuration




```
interface Loopback 0
 ip address 10.0.0.11
pce
 address ipv4 10.0.0.11
 segment-routing
  traffic-eng
   peer ipv4 10.0.0.1
   !
...
router ospf 1
 distribute link-state instance-id 3[x]
```

Distribute IGP LS info into BGP-LS 1

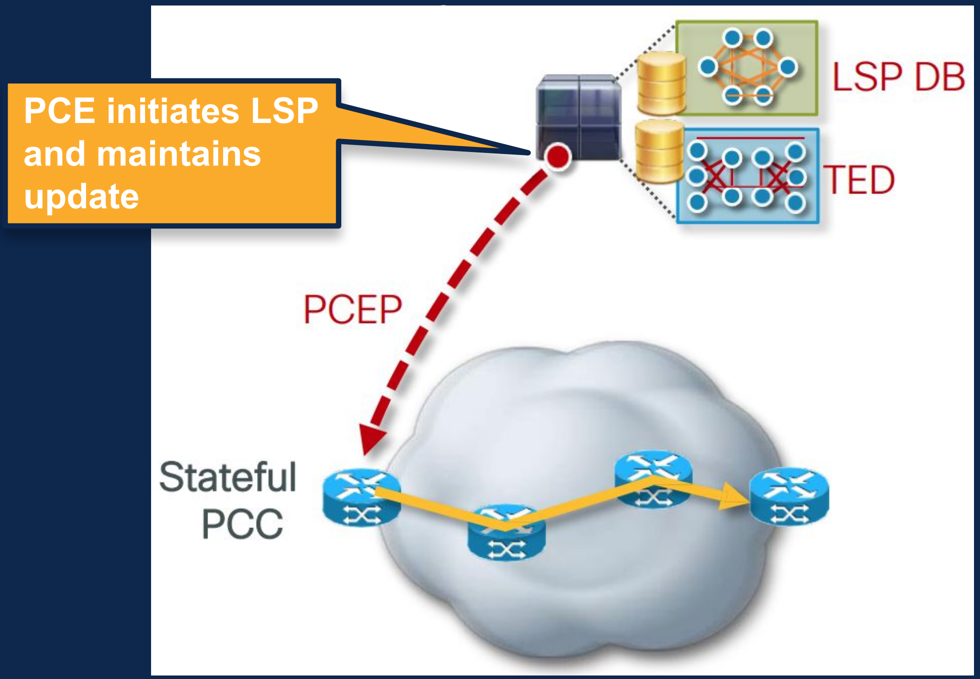
instance-id needed for multi-domain (one instance ID per domain)

```
router bgp 65000
 address-family ipv4 unicast
 !
 address-family link-state link-state
 !
 neighbor 10.0.0.22
  address-family link-state link-state
```

BGP-LS peering 2

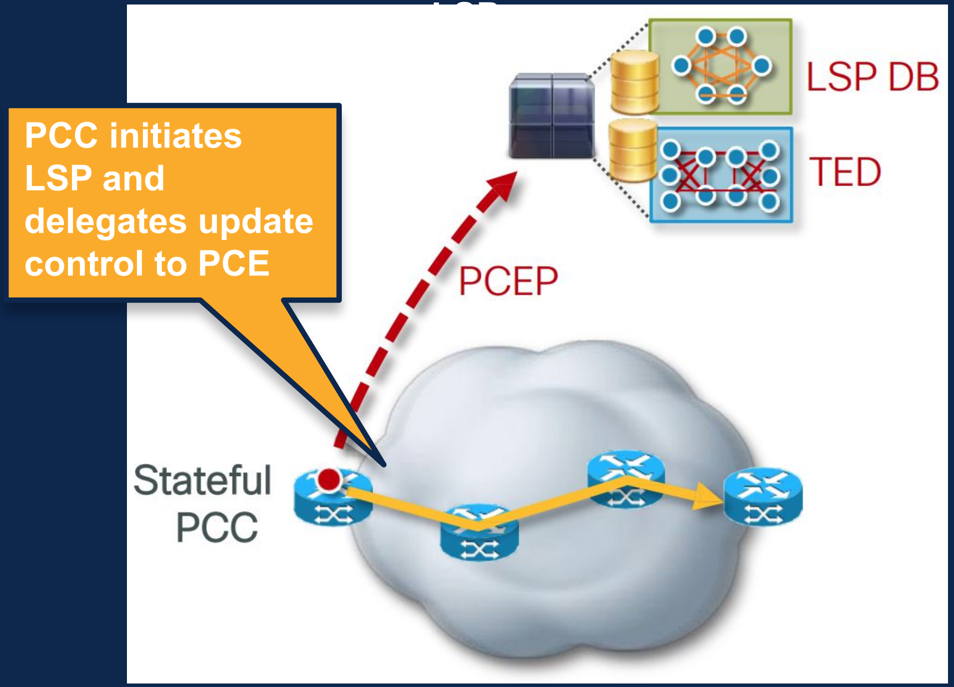
PCE-Initiated and PCC-Initiated LSP

PCE-Initiated



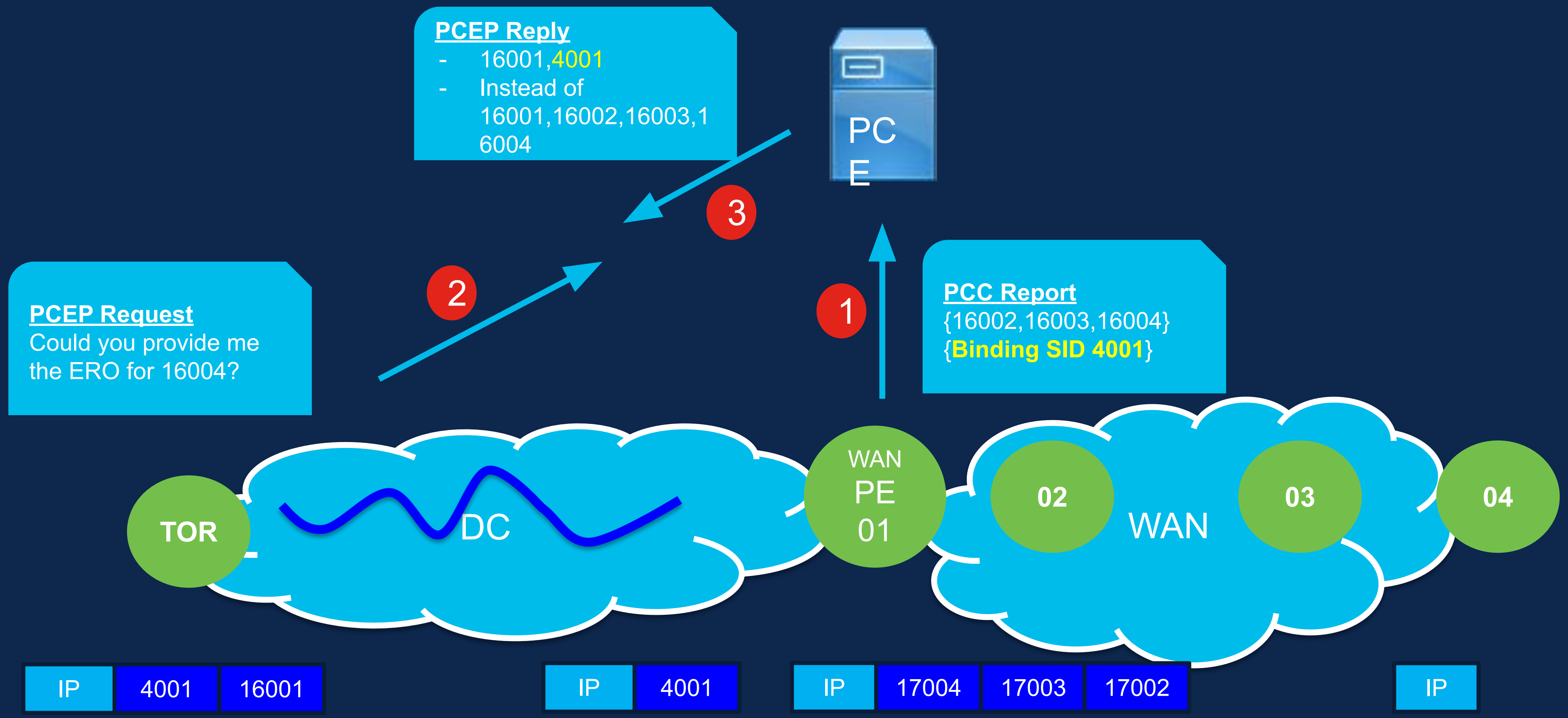
- PCE part of controller managing full path lifecycle
- Tighter integration with application demands

PCC-Initiated

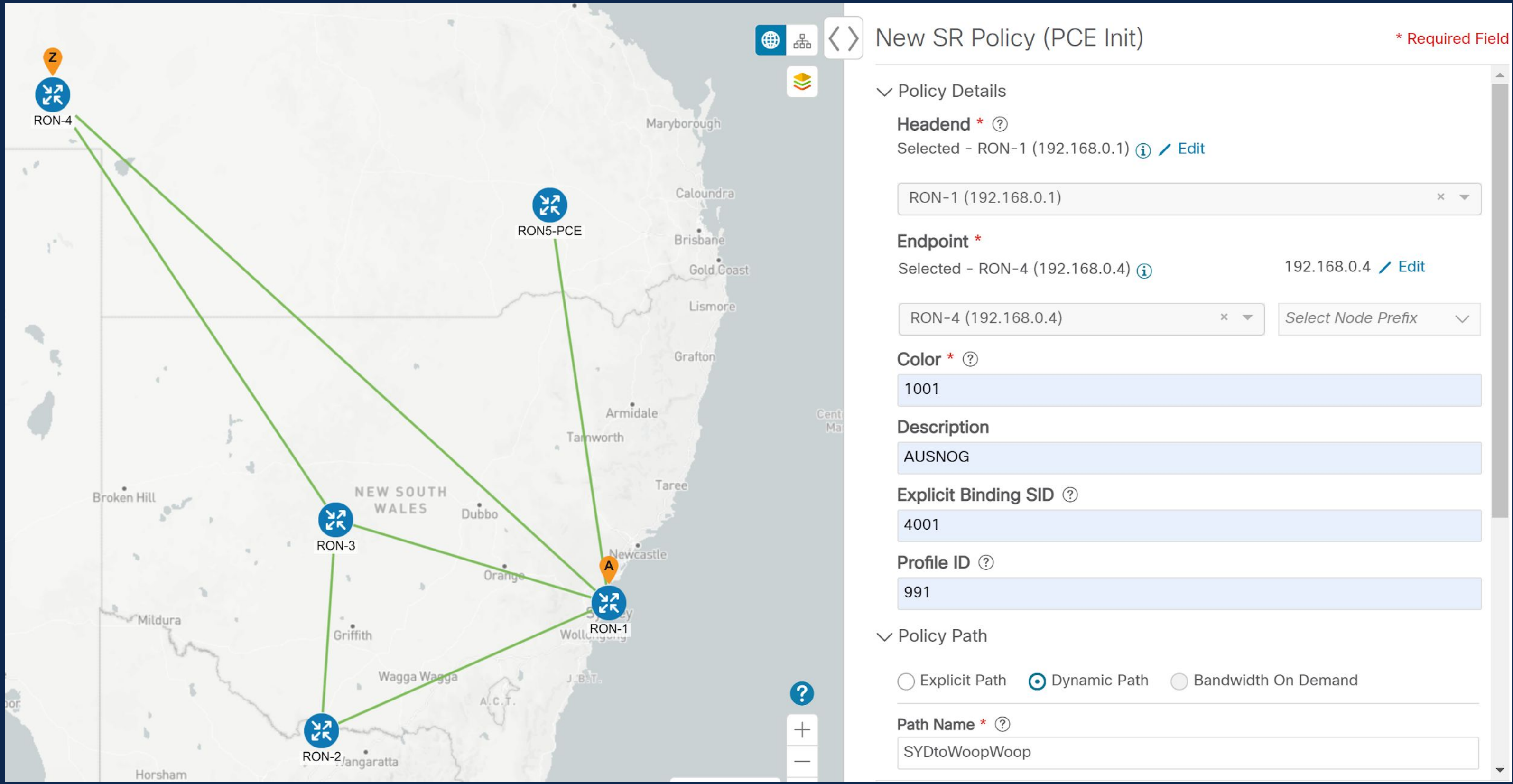


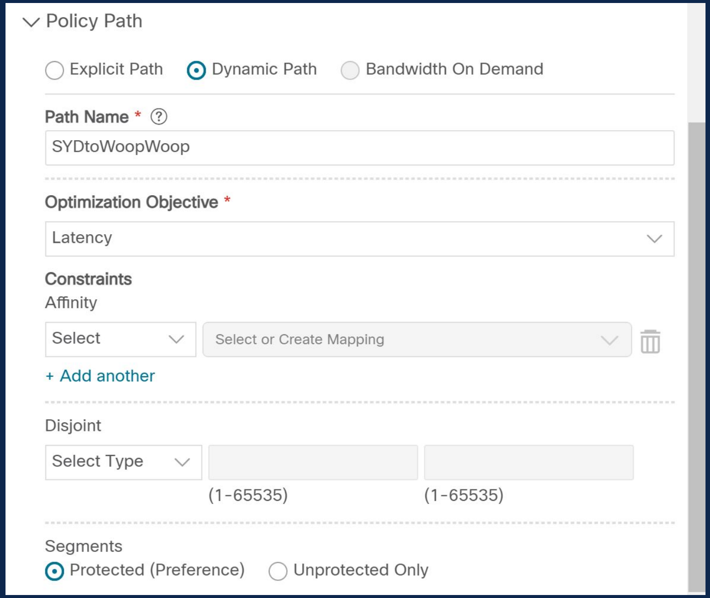
- PCC initiate path setup based on distributed network state
- Can be used in conjunction with PCE-initiated paths

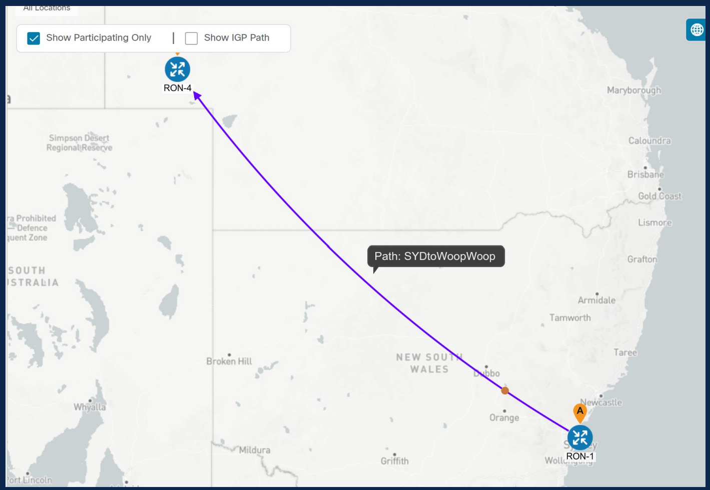
Binding-SID use-case examples



True Programmability







Desire from the Network

- **Simplicity**
  Less numbers of protocols to operate & troubleshoot
  Less numbers of protocol interactions to deal with
  Deliver automated FRR for any topology

- **Scale**
  Avoid thousands of labels in LDP database
  Avoid thousands of MPLS Traffic Engineering LSP’s in the network
  Avoid thousands of tunnels to configure

- **Leverage all services supported over MPLS today (L3/L2 VPN, TE, IPv6)**
  Requires evolution and not revolution

- **Bring the network closer to the applications**

THANK YOU

CISCO

The bridge to possible