


SR-TE Segment List optimizations

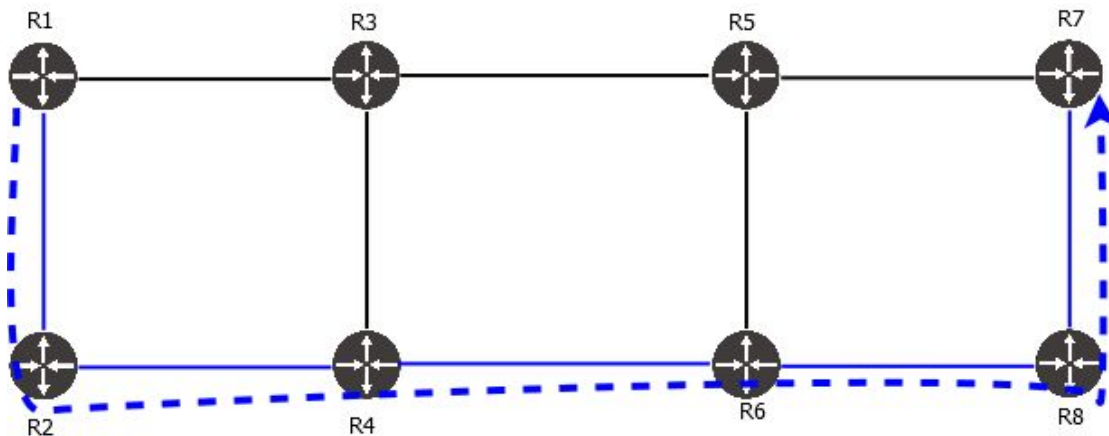
Dmytro Shypovalov
dmytro@vegvisir.ie

About me & who is this session for

- Founder of Vegvisir Systems, developer of SDN controller Traffic Dictator
 - Previously - Network Engineer, working for ISP and vendors
 - TD is a user-friendly SR-TE controller built for network engineers
 - See <https://vegvisir.ie/> for more information
-
- Target audience - engineers who work with SR and have to understand how the SR implementation they use solves the SL generation problem (and its caveats)
 - Or just folks interested in technical details of how SR works
 - Router or controller developers who want to optimize their SR-TE algorithms
 - The goal of the session is to raise awareness about this problem and show how different SR implementations generate wrong segment lists in some cases
- 

What is Traffic Engineering

- Traffic engineering is steering traffic over a path different than the shortest path
- I want to steer traffic from R1 to R7 using only blue links
- R1 (or controller) prunes non-blue links from IGP topology and calculates CSPF
- The result is the list of links: [R1-R2, R2-R4, R4-R6, R6-R8, R8-R7] - in RSVP, this is encoded as Explicit Route object (ERO)
- With classical MPLS-TE, R1 signals an MPLS LSP using RSVP to steer traffic via this path
- Path must be signaled and maintained before any traffic can be forwarded



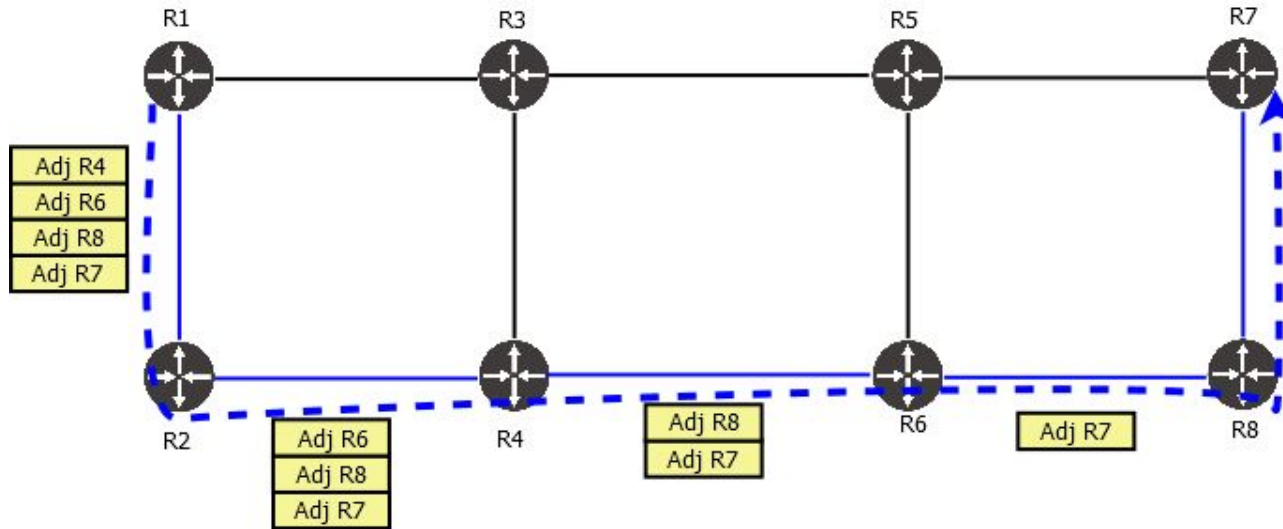
Segment routing - stateless TE

- Path signaling and state maintenance in MPLS-TE leads to complexity, poor scalability and interoperability
- Segment Routing is stateless - desired path is encoded in the label stack itself
- SR is simpler, more scalable, supports ECMP and anycast
- Challenge - compute the label stack (or *segment list*) to forward traffic correctly
 - Wrong segment list will lead to traffic forwarded via a wrong path, or blackholed
 - Suboptimal segment list (more labels than required) will trigger platform limitations (some chips can push 3-4 labels max) or MTU issues
 - Multiple segment lists for ECMP lead to a higher HW resource usage
- Not every SR implementation can generate segment lists
 - An SDN controller can help with that, so router implementations can be quite minimalistic (e.g. FRR, Arista EOS)



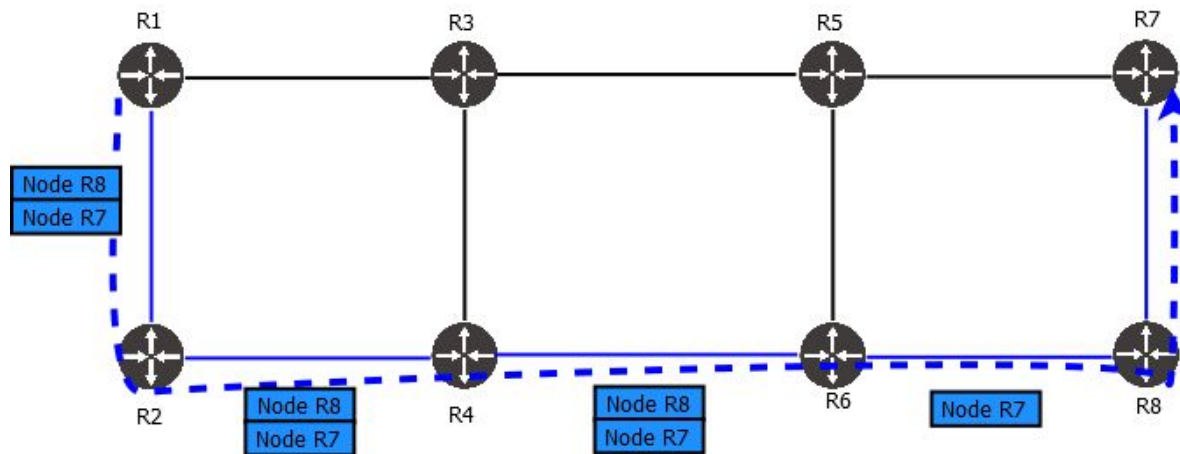
Segment Routing - naive implementation

- Naive approach: for each link in ERO except the first one, push an adjacency segment
- This can result in an absurd amount of labels pushed on the packet - leading to platform limitations and MTU issues



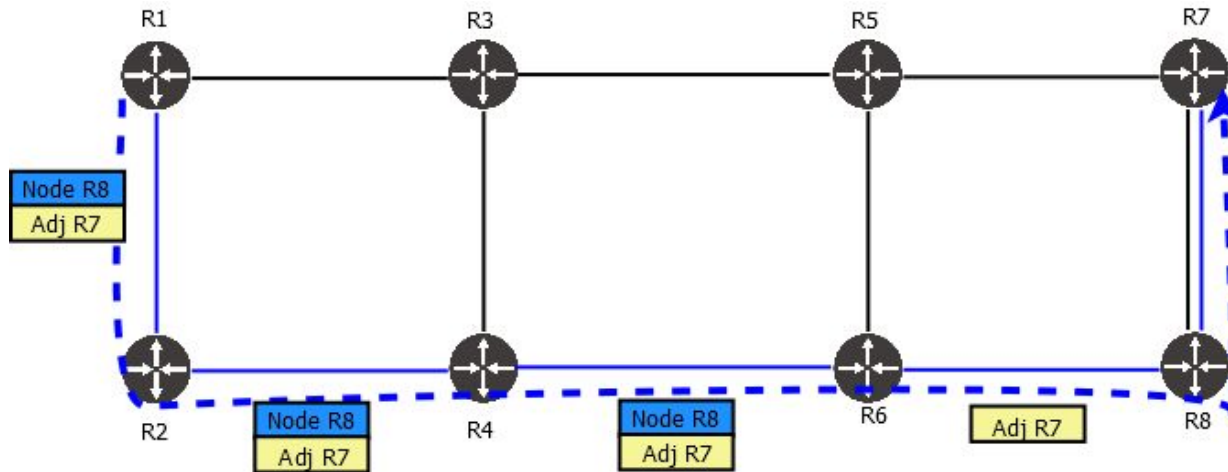
Improvement: prefix SID

- Use prefix SID to reduce the label stack: first steer traffic from R2 to R8 using shortest path, then steer to R7
- Even in large topologies, most TE functions need no more than 2-3 labels, so we won't hit platform limitations or MTU



Adjacency and prefix SID together

- Sometimes, a combination of adj SID and prefix SID is required to steer traffic
- E.g.: steer to R8 following shortest path, then use a specific link to R7



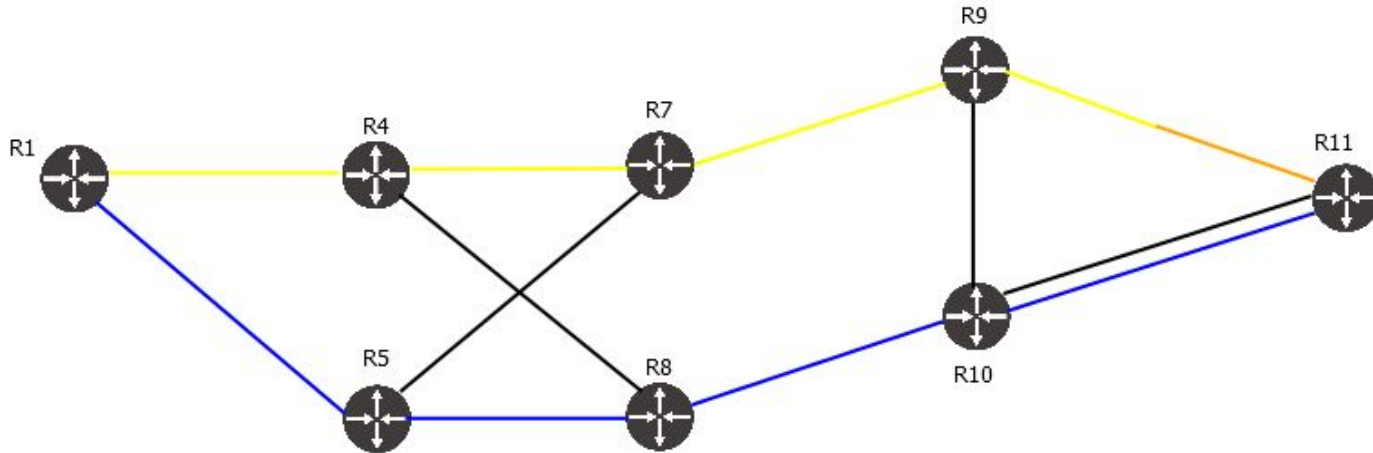
Algorithm basics

- Run CSPF to get ERO (similar to RSVP-TE, except also ECMP is possible)
- Go backwards from the last to the first node and compare non-constrained SPF ERO to the endpoint with CSPF ERO
- When required, add relevant node or adjacency SID, and use this “anchored node” for next SPF-to-CSPF comparison
- There are some caveats with SRGB, ECMP, Global adj SID and Anycast SID
- See blog post <https://routingcraft.net/generating-an-optimal-segment-list-for-sr-te/>



Adjacency and prefix SID together - tests

- Send traffic from R1 to R11 via BLUE links - expecting segment list [Node R5, Node R10, Adj R11]
- Since R5 is directly connected to R1, node SID of R5 is used just for nexthop resolution and not sent in dataplane



Adjacency and prefix SID together - outputs

- TD, IOS-XR, JUNOS calculate the same segment list
- Differences in output due to different SRGB - ignore that

```
TD#show traffic-eng policy R1_R11_BLUE_ONLY_IPV4 detail
```

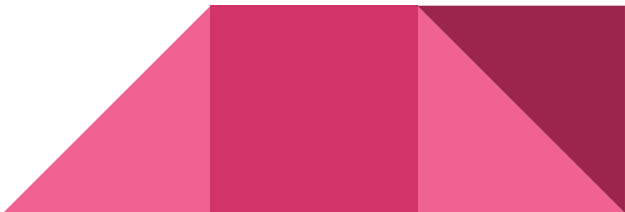
```
-----  
Segment lists:  
[16005, 16010, 24013]
```

```
RP/0/RP0/CPU0:XR#show segment-routing traffic-eng policy
```

```
-----  
SID[0]: 16005 [Prefix-SID, 5.5.5.5]  
SID[1]: 16010 [Prefix-SID, 10.10.10.10]  
SID[2]: 24013 [Adjacency-SID, 10.100.15.10 - 10.100.15.11]
```

```
admin@JUNOS# run show spring-traffic-engineering lsp detail
```

```
-----  
computed segment : 1 (computed-node-segment):  
  node segment label: 21  
  router-id: 5.5.5.5 ::1  
computed segment : 2 (computed-node-segment):  
  node segment label: 16010  
  router-id: 10.10.10.10 ::1  
computed segment : 3 (computed-adjacency-segment):  
  label: 24013  
  source router-id: 10.10.10.10, destination router-id:  
11.11.11.11  
  source interface-address: 10.100.15.10, destination  
interface-address: 10.100.15.11
```



Global adjacency SID

- Same topology and constraint as before
- Global adj SID can be used to optimize the segment list from 3 to 2 SID in this example (instead of node R10 + adj R11, just use R10 global adj SID towards R11)
- This is also similar to End.X function in SRv6 (SRv6 has no analog to local adj SID)
- Global Adj SID is advertised as index (just like prefix SID), so the implementation should correctly add index to the SRGB of the relevant node
- If not supporting global adj SID - just ignore it and use local adj SID (if available)
- Example on TD:

```
TD#show traffic-eng policy R1_R11_BLUE_ONLY_IPV4 detail
```

```
-----  
Segment lists:  
[900005, 902100]
```



Global adjacency SID (cont.)

- JUNOS puts label equal to global adj SID index (without SRGB) - garbage label for which there is no forwarding entry on any router (so traffic will be dropped)
- IOS-XR is confused by global adj SID and puts corrupt label u32::MAX in the label stack
 - CEF is unresolved and traffic is dropped on headend
- Both behave this way even when there is another local adj SID available

```
admin@JUNOS# run show spring-traffic-engineering lsp detail
```

```
-----  
computed segments count: 3  
  computed segment : 1 (computed-node-segment):  
    node segment label: 21  
    router-id: 5.5.5.5 ::1  
  computed segment : 2 (computed-node-segment):  
    node segment label: 900010  
    router-id: 10.10.10.10 ::1  
  computed segment : 3 (computed-adjacency-segment):  
    label: 2100  
    source router-id: 10.10.10.10, destination router-id: 11.11.11.11  
    source interface-address: 10.100.15.10, destination  
interface-address: 10.100.15.11
```

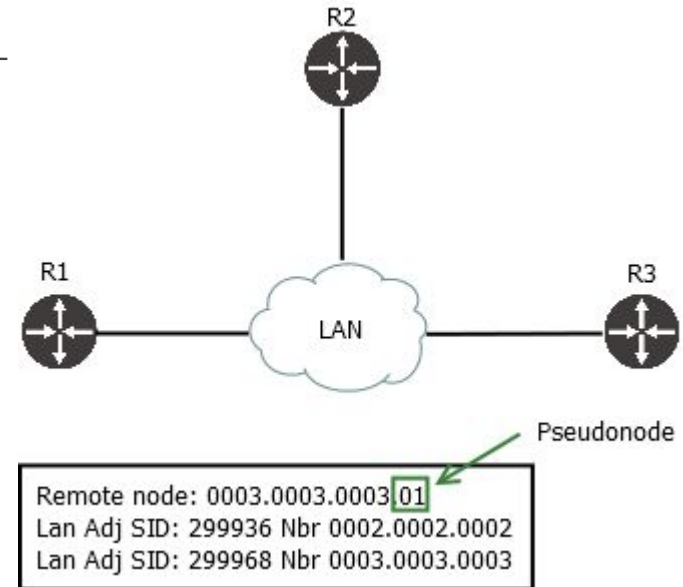
```
RP/0/RP0/CPU0:IOSXR#show segment-routing traffic-eng policy
```

```
-----  
SID[0]: 16005 [Prefix-SID, 5.5.5.5]  
SID[1]: 900010 [Prefix-SID, 10.10.10.10]  
SID[2]: 4294967295 [Adjacency-SID, 10.100.15.10 - 10.100.15.11]
```



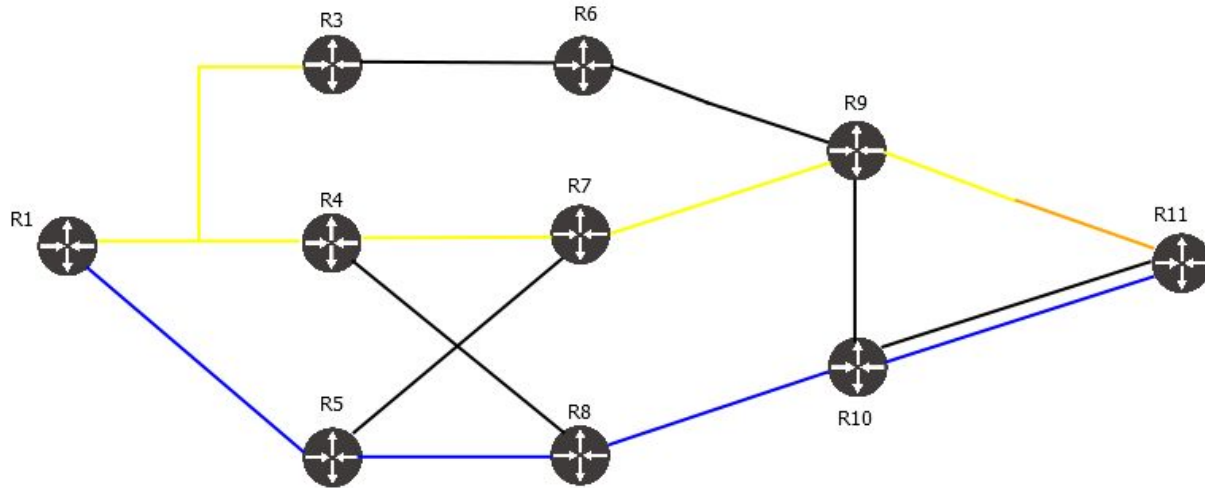
Broadcast links

- In a typical ISP network, most links are configured as point-to-point
- However, IS-IS and OSPF also support broadcast links - i.e. more than 2 routers on a LAN
- Common to have broadcast links as misconfiguration
- Routers elect a DIS which generates pseudonode LSP
- While regular IS-IS LSP would have Adj SID TLV associated with a neighbor system ID; the LSP describing pseudonode connection has multiple LAN Adj SID (per neighbor)
- SL generation algorithm should correctly handle both P2P and broadcast links and use the relevant Adj SID or LAN Adj SID



Broadcast links - test topology

1. Send traffic from R1 to R11 using only YELLOW links (using LAN between R1 and R4)
2. Send traffic from R1 to R11 using only BLUE links - see if the algorithm gets confused by presence of a LAN segment in the topology



Broadcast links - tests

- TD and JUNOS both generate correct segment lists, although slightly different in first test (TD returns [R4, R7, R11] and JUNOS [R4, R9, R11])
- IOS-XR fails the path in the first test, and computes an incorrect segment list in the second test

```
TD#show traffic-eng policy R1_R11_YELLOW_ONLY_IPV4
detail
```

```
-----
Segment lists:
  [16004, 16007, 16011]
```

```
RP/0/RP0/CPU0:XR#show segment-routing traffic-eng policy
```

```
-----
Affinity:
  include-all:
    YELLOW
Dynamic (inactive)
Last error: No path found
```

```
admin@JUNOS# run show spring-traffic-engineering lsp detail
```

```
-----
computed segments count: 3
  computed segment : 1 (computed-node-segment):
    node segment label: 20
    router-id: 4.4.4.4 ::1
  computed segment : 2 (computed-node-segment):
    node segment label: 16009
    router-id: 9.9.9.9 ::1
  computed segment : 3 (computed-node-segment):
    node segment label: 16011
    router-id: 11.11.11.11 ::1
```

Broadcast links - tests (cont.)

- The second test on XR will result in traffic being ECMP'ed over R4/R5, thus violating the “BLUE only” constraint

```
RP/0/RP0/CPU0:XR#show segment-routing traffic-eng policy
```

```
-----
```

```
Affinity:
```

```
include-all:
```

```
BLUE
```

```
SID[0]: 16010 [Prefix-SID, 10.10.10.10]
```

```
SID[1]: 24005 [Adjacency-SID, 10.100.15.10 - 10.100.15.11]
```

```
RP/0/RP0/CPU0:XR#show segment-routing traffic-eng forwarding policy name srte_c_1101_ep_11.11.11.11
```

```
-----
```

```
Color: 1101, End-point: 11.11.11.11
```

```
Preference: 100 (configuration)
```

```
Name: R1_R11_BLUE_ONLY_IPV4
```

```
Paths:
```

```
Path[0]:
```

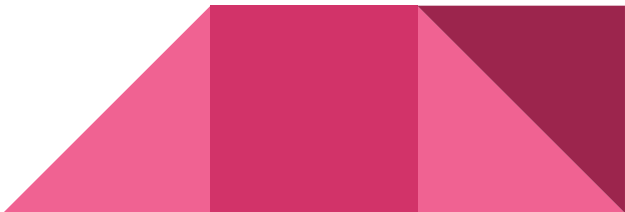
```
Outgoing Interfaces: GigabitEthernet0/0/0/1
```

```
Label Stack (Top -> Bottom): { 16010, 24005 }
```

```
Path[1]:
```

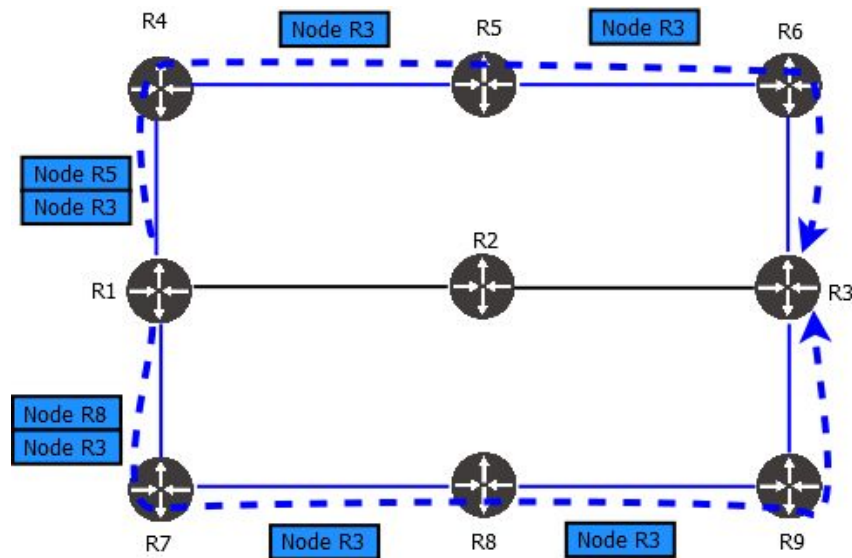
```
Outgoing Interfaces: GigabitEthernet0/0/0/2
```

```
Label Stack (Top -> Bottom): { 16010, 24005 }
```



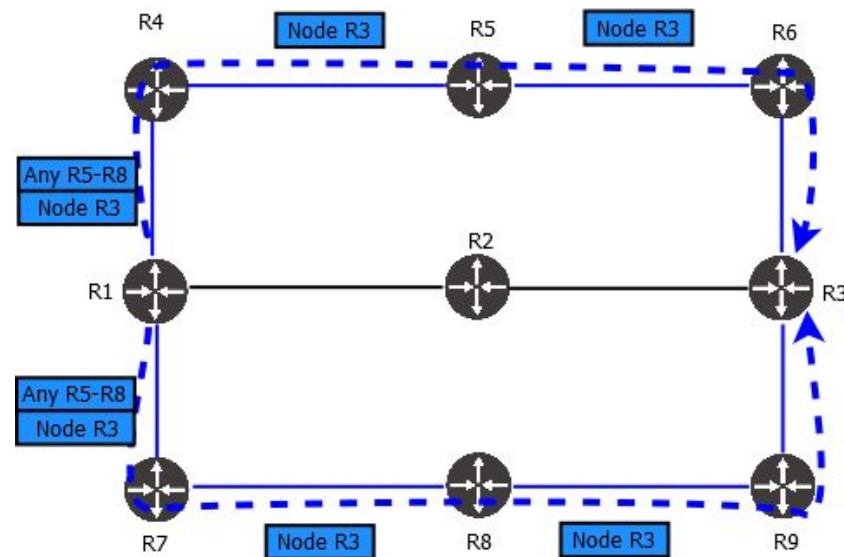
ECMP

- Segment Routing is ECMP-aware, so must be SR-TE!
- If we want to steer traffic from R1 to R3 using blue links, 2 segment lists are required: [R5, R3] and [R8, R3]
- Multiple SL cause higher HW resource usage
- If a controller is used to advertise policies, ECMP can become a problem
 - BGP-SRTE supports multiple SL
 - PCEP needs to support [\[draft-ietf-pce-multipath\]](#)
 - BGP-LU needs add-path



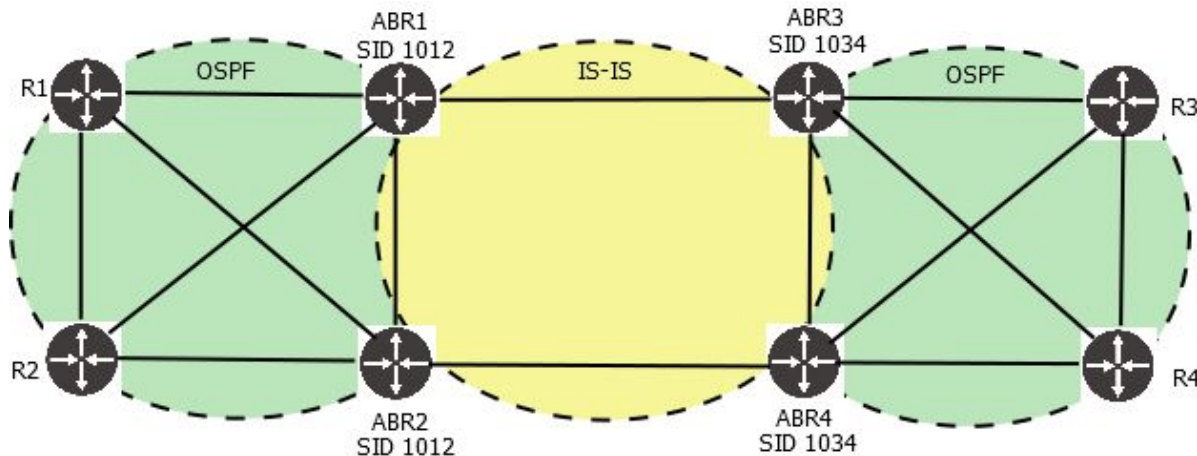
Anycast SID

- Configure the same prefix SID on R5 and R8
- Now we can steer traffic into ECMP with just one segment list, using anycast SID
- SL generation algorithm must do a number of checks to make sure the anycast SID can be used:
 - SRGB must match
 - No other nodes must have the same SID
 - After anycast SID, path must converge on one node or there must be another set of anycast nodes (but no adj SID is allowed)



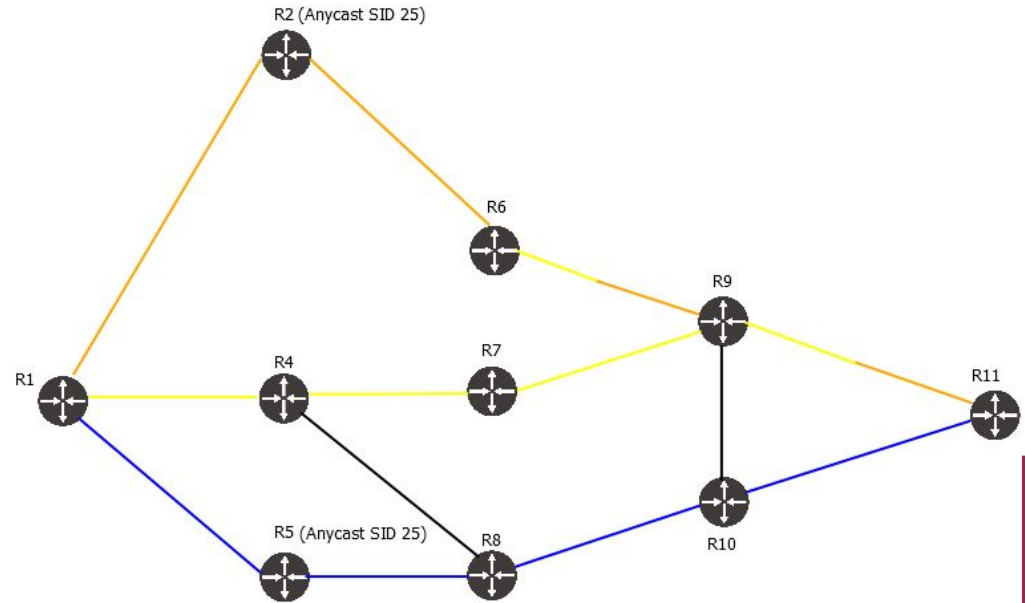
Anycast SID in multi-domain topologies

- Anycast SID is especially useful in multi-domain topologies, as it provides load balancing and resiliency
- From the controller perspective we see BGP-LS topology but not the IGP configuration, so it's safer to assume there is no redistribution/leaking between IGP
- Hence, in multi-domain SR-TE, always add ABR anycast SID to the segment list
- Check that after anycast SID, there is another anycast SID or the path converges on one node



ECMP and Anycast SID - test topology

- Constraint: BLUE or ORANGE links
- Expecting segment lists when anycast SID is not enabled on R2 and R5
- Expecting optimization to one SL when anycast is enabled



ECMP and Anycast SID - tests

- TD generates 2 SL: [R2, R11] and [R5, R11] and optimizes them to [Anycast R2-R5, R11]

!! without anycast SID

```
TD1#show traffic-eng policy R1_R11_BLUE_OR_ORANGE_IPV4 detail
```

Segment lists:

[16005, 16011]

[16002, 16011]

!! with anycast SID

```
TD1#show traffic-eng policy R1_R11_BLUE_OR_ORANGE_IPV4 detail
```

Segment lists:

[16025, 16011]



ECMP and Anycast SID - tests (cont.)

- JUNOS generates 2 SL: [R2, R11] and [R6, R11] - different from TD but still correct
- Doesn't optimize with anycast SID

```
admin@JUNOS# run show spring-traffic-engineering lsp detail
```

```
-----  
Total number of computed paths: 2
```

```
Segment ID : 128
```

```
Computed-path-index: 1
```

```
  computed segments count: 2
```

```
    computed segment : 1 (computed-node-segment):
```

```
      node segment label: 22
```

```
      router-id: 6.6.6.6 ::1
```

```
    computed segment : 2 (computed-node-segment):
```

```
      node segment label: 16011
```

```
      router-id: 11.11.11.11 ::1
```

```
Segment ID : 129
```

```
Computed-path-index: 2
```

```
  computed segments count: 2
```

```
    computed segment : 1 (computed-node-segment):
```

```
      node segment label: 21
```

```
      router-id: 5.5.5.5 ::1
```

```
    computed segment : 2 (computed-node-segment):
```

```
      node segment label: 16011
```

```
      router-id: 11.11.11.11 ::1
```



ECMP and Anycast SID - tests (cont.)

- IOS-XR doesn't support ECMP with SR-TE and generates just one segment list [R6, R11]
- Anycast SID doesn't work either (I think ECMP support is a prerequisite for anycast SID)
- There is a command **"anycast-sid-inclusion"** but it doesn't work
 - One effect I noticed this command has is that it forces the algorithm to prefer prefix SID without N flag in segment list (by default, only prefix SID with N flag are used)
 - However, if a prefix SID is advertised by more than one router (i.e. anycast), XR can't use that SID

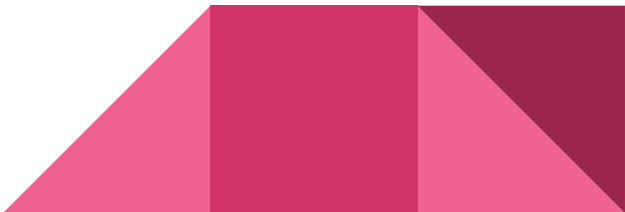
```
RP/0/RP0/CPU0:XR#show segment-routing traffic-eng policy
```

```
-----
```

```
Metric Type: TE,    Path Accumulated Metric: 2000
```

```
  SID[0]: 16006 [Prefix-SID, 6.6.6.6]
```

```
  SID[1]: 16011 [Prefix-SID, 11.11.11.11]
```



Choosing which SID to use

- What if a router has multiple prefix SID configured?
- Prefix SID can have N (node flag) which means it's unique to this node (like a router-id)
 - Prefix SID without N flag can be configured on multiple nodes
- Assuming no anycast is used (we want to steer traffic through THIS node), a SID with N (node) flag should be preferred
- Ideally not to use SID with explicit null flag (because SR-TE has its own explicit null mechanism, using SID with exp-null can result in multiple exp-null labels imposed)
- TD chooses prefix SID in the following order:
 - Node SID without exp-null
 - Prefix SID without exp-null
 - Node SID with exp-null
 - Prefix SID with exp-null



Choosing which SID to use (cont.)

- IOS-XR chooses node SID (N flag must be set) with the lowest IP; SID without N flag are ignored
- With “anycast-sid-inclusion” configured under policy, XR prefers prefix SID (without N flag) - unless this SID is configured on more than one node - strange behaviour!
- JUNOS always picks prefix SID equal to the router-id
 - Doesn't check for N flag
 - If the router-id loopback doesn't have prefix SID, the policy will fail
 - For IPv6 SR-MPLS (not SRv6!) policies, JUNOS requires IPv6 router-id



Conclusion

- Segment Routing simplifies traffic engineering for the network operator - but not for the developer!
- CSPF algorithm from RSVP-TE is not suitable to be reused for SR-TE: no ECMP, no anycast, no EPE
- A good CSPF and Segment list generation algorithm should support ECMP and anycast by design
- When deploying SR-TE, test how your implementation of choice generates segment lists in different scenarios. Things to watch: ECMP, broadcast links, global adj SID, selection among multiple SID

