# Source Routing 2.0

## Why Now?  Why Again?

Nick Slabakov

*slabakov@juniper.net*

v3

# Outline

Source Routing

- Historical Notes

SPRING

- Principles of operation
- Why it has motivated new discussions on source routing

SPRING Inspirations

- SPRING-inspired second look at existing problems

Conclusions

# Terminology Level-Set

- Source Routing
  - Explicit definition of a packet path within the packet header by the source.
  - Source Routing is a generic term, there are many methods of doing it.
- Segment Routing
  - Emergent network architecture based on the distribution of label (and IPv6 segment) info in the IGP.
  - Segment Routing is one specific way of doing Source Routing.
- SPRING (Source Packet Routing In NetworkinG)
  - IETF working group tasked with standardizing the architecture and protocols associated with Segment Routing.

# Source Routing – Short History

## Key idea

- Prescribe the path of the packet in its header at the source; the source has unique knowledge about the desired path.
  - A nice side-effect is that loops can be avoided.
- Reduce/remove forwarding state in the network, put it in the packet instead.

## Examples

- Niche high-performance interconnects
  - Myrinet, SpaceWire, etc.
- Token Ring, APPN,  ANR (IBM), …
- IP
  - IPv4 – LSRR and SRRR options.
  - IPv6 – Extension header of routing type.

# IP Header-Based Source Routing

## Security Concerns and Solutions

IPv4 options and IPv6 header extensions

- Treated as easily spoof-able and prone to amplification attacks.
- Generally disabled on all Internet-connected routers.
- RFC5095 actually deprecates Type 0 routing extension header:
  - *“An IPv6 node that receives a packet with a destination address assigned to it and that contains an RH0 extension header MUST NOT execute the algorithm specified in the latter part of Section 4.4 of [RFC2460] for RH0 …”.*

Tunneling

- Tunneling at the SP edge delineates the trust boundary.
- Tunneling is a common method of doing source routing from the SP edge
  - E.g. MPLS/RSVP uses EROs extensively, but operates under the operator’s sphere of control.

# Why Now?  Why Again?

## SPRING (a.k.a. Segment Routing)

- Tunnel packet from source to destination by encoding the path in the tunnel header of the packet
  - Combine the benefits of source routing and tunneling.
- The more you care about describing the specific path, the more state you need to insert in the header
  - Conversely, if you don’t care about the specific path, less state is needed.

## Centralized Controllers

- Itself not a new idea, but one with new blood in it.
  - Every SDN has one ☺
- Path calculation and path programming – on routers and on hosts.

# Outline

Source Routing

- Historical Notes

SPRING

- Principles of operation
- Why it has motivated new discussions on source routing

SPRING Inspirations

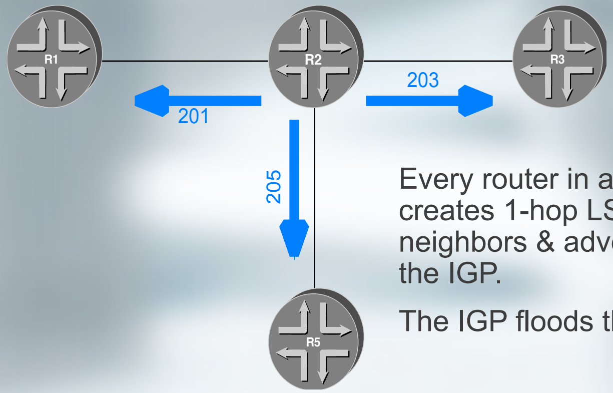
- SPRING-inspired second look at existing problems

Conclusions

# Key Concepts:

## The Two Building Blocks of SPRING

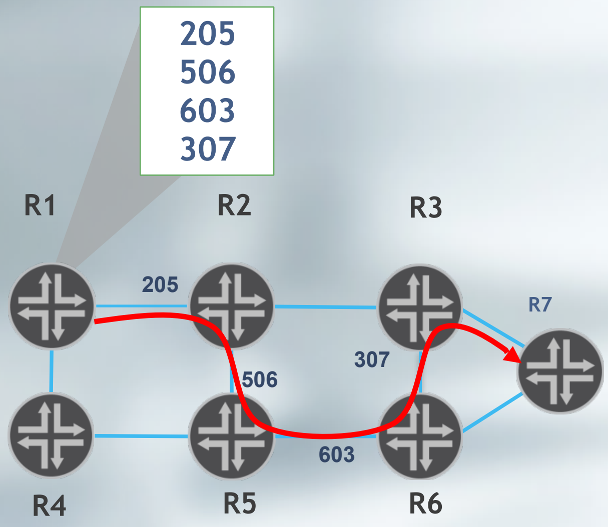
R2 Area 0 advertisement:
Local Label 201, To 192.168.1.1
Local Label 203, To 192.168.1.3
Local Label 205, To 192.168.1.5



Every router in an IGP domain creates 1-hop LSP to its IGP neighbors & advertises the label in the IGP.

The IGP floods the labels.

1. Advertising Labels in the IGP*



Ingress Router uses a stack of labels to describe a path. The label stack is the ERO.

Each router POPs the top label and forwards the rest.

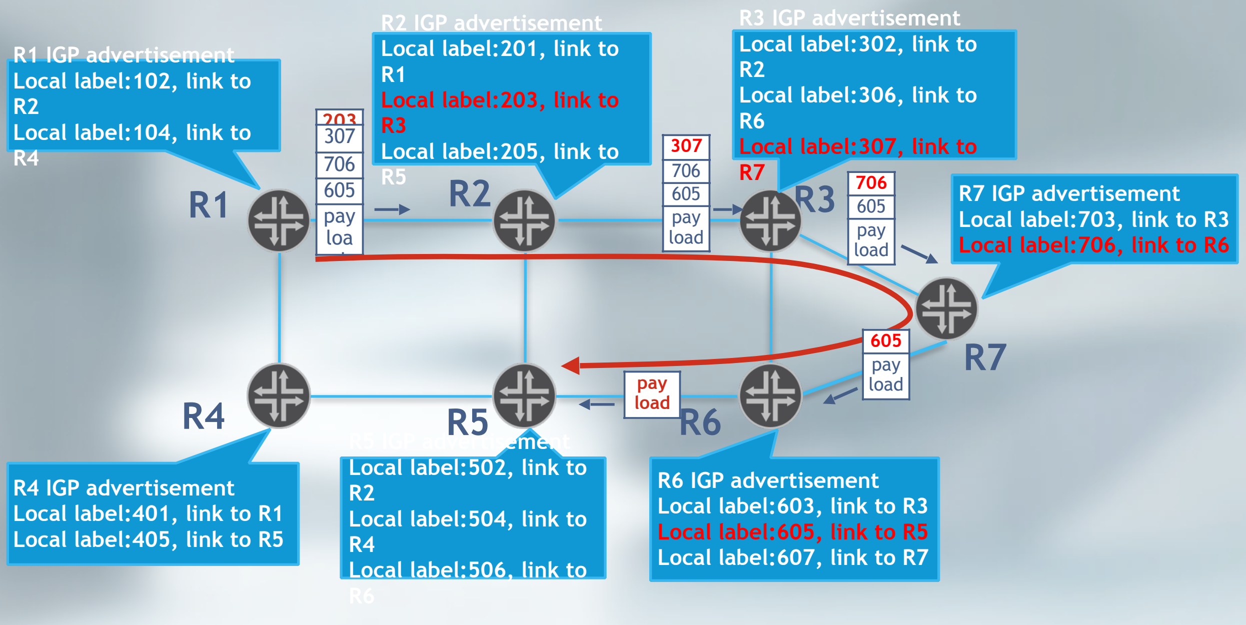
Accomplishes explicit routing without signaling forwarding state.

2. Forwarding based on a stack of MPLS labels**

* For some data-center use-cases, there are proposals to utilize BGP for the same purpose.

** There is an IPv6 data-plane proposal for SPRING, but the concepts are similar.

# SPRING: Adjacency Label



R1 IGP advertisement
Local label:102, link to R2
Local label:104, link to R4

R2 IGP advertisement
Local label:201, link to R1
Local label:203, link to R3
Local label:205, link to R5

R3 IGP advertisement
Local label:302, link to R2
Local label:306, link to R6
Local label:307, link to R7

R7 IGP advertisement
Local label:703, link to R3
Local label:706, link to R6

R4 IGP advertisement
Local label:401, link to R1
Local label:405, link to R5

R5 IGP advertisement
Local label:502, link to R2
Local label:504, link to R4
Local label:506, link to R6

R6 IGP advertisement
Local label:603, link to R3
Local label:605, link to R5
Local label:607, link to R7

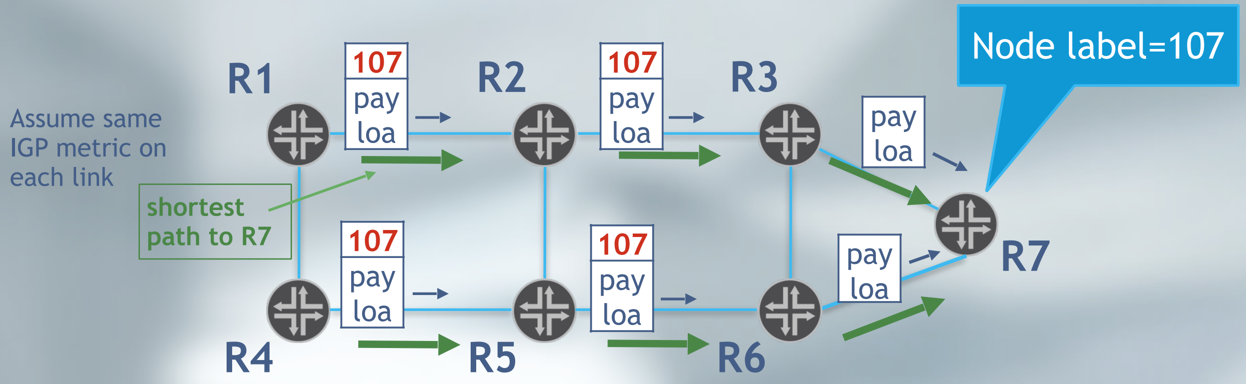
## Observations:

- Amount of State:
  - No LSPs or per-LSP state on transit routers. That is nice.
  - Then again, if you want per-LSP stats, or TE, or bandwidth reservation, it is not so nice.
- Trivial method of forwarding
  - It requires deep label stack support (mitigated by node-segments).
  - There are practical challenges in imposing such deep stacks in both custom and merchant silicon.
- We almost never care to describe the path with such specificity
  - E.g. “loose-hop” is often sufficient.

- To send a packet to R5 along the path (R2,R3,R7,R6), R1 sends to packet to R2 with label stack = <203,307,706,605>.
- Each router determines next-hop from top label, then POPs the label.

# SPRING: Node Label (SID)

## Global Node Label Version



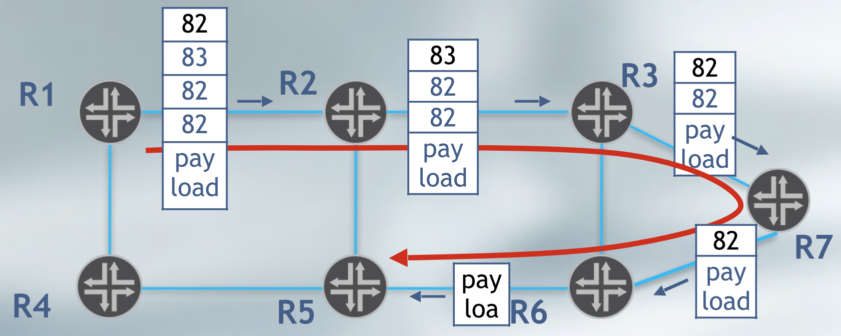
See backup slide for discussion on the solution to the global label problem.

- In simplest version, each router advertises a global node label in the IGP.
- Whenever a router receives a packet with label=107, it forwards the packet (without modifying the label) along the shortest path to R7.
- **Problem:** Global node label is not compatible with the local label assignment used by MPLS protocol suite (RSVP, LDP, BGP-LU, etc.)
  - In MPLS, a router decides the values of the labels that other routers use to send it traffic.
  - What if R6 has already used label=107 to advertise a FEC-label binding in LDP?

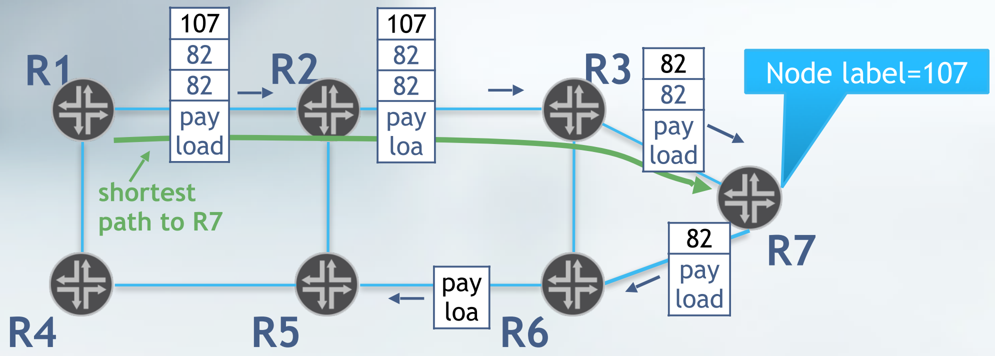
# Label Stack “Compression”

## Using Both Adjacency and Node Labels

Using only adjacency labels requires 4 label stack for explicit path.



Can shorten label stack by 1 using a node label to get to R7  (and 2 more labels to get to R5).



# Other Segments You Might Encounter

Prefix and Anycast SIDs

- Superset of Node segment, have global significance.

PeerNode, PeerAdj, PeerSet

- For egress peer engineering use-cases.

Mapping Servers

- To facilitate interoperability with LDP.

SID/Label Binding TLV

- Used to associate a label with a FEC and ERO.
- FEC can represent an LSP signaled by another protocol.
- FEC can represent a context-id for egress node protection.

BGP and BGP-LU enhancement work

- De-facto protocol of choice for MSDCs.
- draft-keyupate-idr-bgp-prefix-sid, draft-gredler-idr-bgplu-epe.

# Outline

Source Routing

- Historical Notes

SPRING

- Principles of operation
- Why it has motivated new discussions on source routing

SPRING Inspirations

- SPRING-inspired second look at existing problems

Conclusions

# Useful Concepts from SPRING

## Predictable label values

- Good for troubleshooting
  - If I know the label values along the way, I don’t have to look them up.
- Good for incorporating a controller
  - Controller does not need to read label values, it can simply “know” them, so a few steps can be saved in creating the label stacks that describe paths.

## The notion of a Node SID

- One instruction (label) that takes you from the source to the destination via whatever ECMP path is available between them
- Elegant, powerful, cheap.

So people thought …
Can’t we benefit from these in our existing networks?

REFERENCE: draft-geib-spring-oam-usecase

# SPRING Use-Case #1

## Exhaustive Data-Plane Monitoring using SPRING

- Run “normal” MPLS control and data-plane
- In addition, assign and advertise the following adjacencies:
  - Adj-SID for each single-link interface and for each AE interface
  - Unique Adj-SID per physical links of a AE bundles
  - Node-SIDs
- The Path Monitoring Server (PMS) can now construct arbitrary paths without creating state in the network

Probe examples from PMS to R1

(Assuming the Payload’s destination IP address is the PMS, so the packet can return to it)

| Payload | 301 | 403 | 504 | 205 | 102 |
|---|---|---|---|---|---|

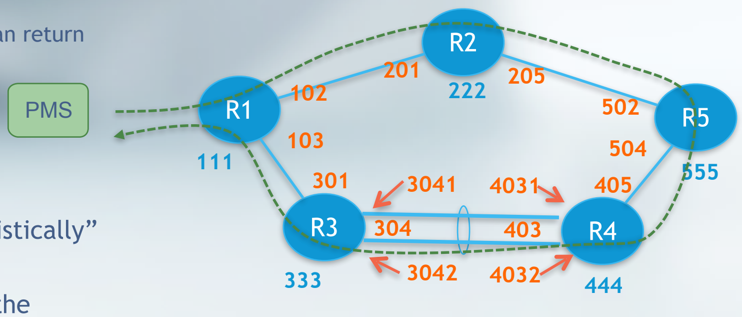
“Traverse the ring clock-wise using hashing on LAGs”

| Payload | 301 | 4031 | 504 | 205 | 102 |
|---|---|---|---|---|---|

“Traverse the ring clock-wise using the upper LAG link deterministically”

| Payload | 301 | 4031 | 444 |
|---|---|---|---|

“Route the probe to R4 via the shortest path (don’t care about the direction), then exercise the upper LAG link”



# SPRINGspiration #1: The Same Use-Case

## Solved With RSVP (1) and Static LSPs (2)

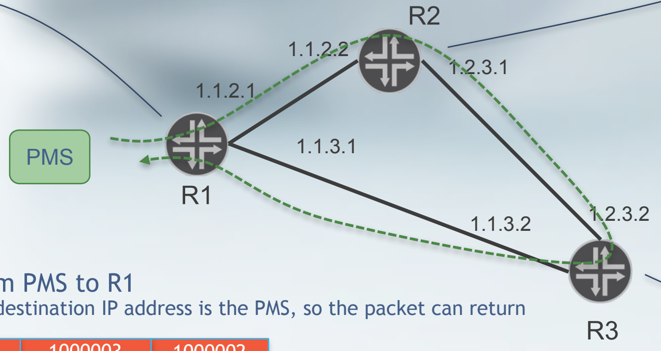
### Exhaustive path monitoring with RSVP

- https://www.nanog.org/meetings/nanog57/presentations/Tuesday/tues.general.GuilbaudCartlidge.Topology.7.pdf
- Create an exhaustive mesh of explicitly routed RSVP LSPs that cover not only the best path, but all paths
- Send OAM probes on all paths, monitor the results, correlate them, and deduce failing links
- That is pretty cool, but creates significant additional per-LSP state in the network, just for OAM traffic

### Exhaustive path monitoring with static LSPs

- Other operators have chosen to use static LSPs between neighboring routers, just to get around that additional RSVP state
- SPRING Concepts: Predictable Labels, POP-and-forward

```
mpls {
      static-label-switched-path R1-
R2{
          transit 1000002 {
               next-hop 1.1.2.2;
               pop;
          }
      }
      static-label-switched-path R1-
R3 {
          transit 1000003 {
               next-hop 1.1.3.2;
               pop;
          }
      }
}
```



Probe examples from PMS to R1

(Assuming the Payload's destination IP address is the PMS, so the packet can return to it)

| Payload | 1000001 | 1000003 | 1000002 |
|---|---|---|---|

"Traverse the ring clock-wise"

```
mpls {
      static-label-switched-path R2-
R1{
          transit 1000001 {
               next-hop 1.1.2.1;
               pop;
          }
      }
      static-label-switched-path R2-
R3 {
          transit 1000003 {
               next-hop 1.2.3.2;
               pop;
          }
      }
}
```

```
mpls {
      static-label-switched-path R3-
R1{
          transit 1000001 {
               next-hop 1.1.3.1;
               pop;
          }
      }
      static-label-switched-path R3-
R2 {
          transit 1000002 {
               next-hop 1.2.3.1;
               pop;
          }
      }
}
```

# SPRINGspiration #2

## Creating MPLS Overlays in the Data-Center

CAUTION:  Controversy

The VM and Server labels are not interesting

- Typically controller-assigned and manages as part of the orchestration
- Only meaningful to hosts, so the network doesn’t care

Egress TOR labels is what we forward on

- How does the Ingress ToR resolve that Egress ToR label?  Ingress ToR is usually not directly connected to the Egress ToR
  - Using SPRING Node-SID
    - » Upgrade to SR needed (or BGP-LU extensions)
  - Using ToR-to-ToR RSVP/LDP mesh
    - » Per-LSP state is in the order of N$^2$
  - Static LSPs
    - » With remote next-hops
    - » And resolution via hop-by-hop RSVP or BGP-LU LSPs
    - » Per-LSP state is in the order of N

Ingress
ToR

Egress
ToR

For a good reasoning on why MPLS in the DC, see:
http://www.slideshare.net/DmitryAfanasiev1/yandex-nag201320131031

# SPRINGspiration #2

## Static LSPs With Stitching

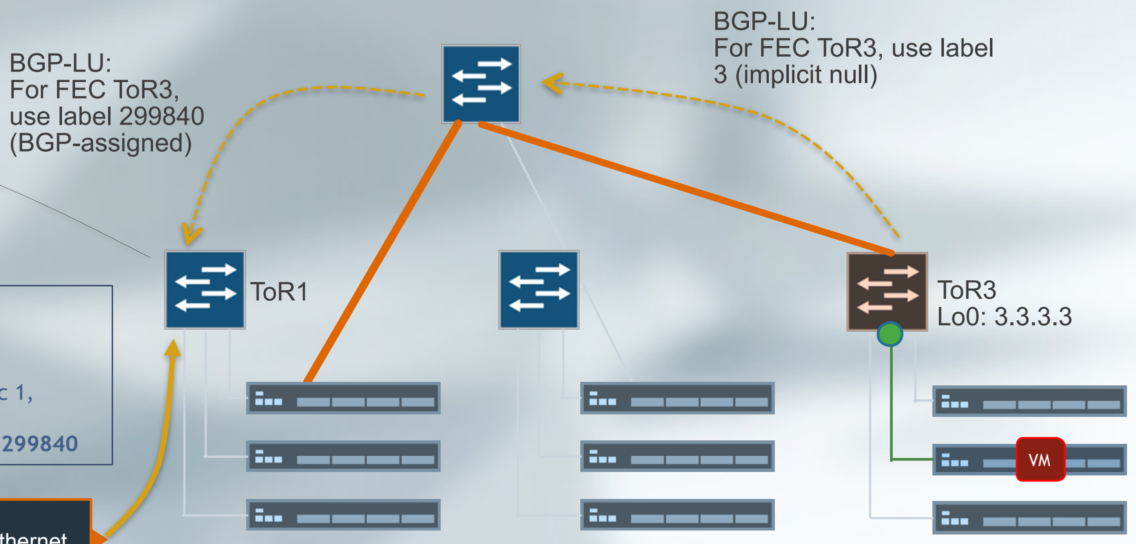
**ToR1 Config (Ingress):**

```
mpls {
      static-label-switched-path TOR1-TOR3 {
            transit 1000003 {
                 next-hop 3.3.3.3;
                 stitch;
            }
      }
}
```

**ToR1 Forwarding:**

```
show route table mpls.0

1000003    *[MPLS/6] 15:38:40, metric 1,
metric2 0
     > to 1.2.3.2 via ge-0/0/1.0, Swap 299840
```



| | | | | |
|---|---|---|---|---|
| Customer packet | MPLS label VM | MPLS label Egress server | 1000003 | Ethernet |

## Benefits

- Retain predictable label assignments for ToRs (ToR3 is always addressed with label 1003 by everyone – good for troubleshooting
  - Just like Node-SID from the server perspective ☺
- Use existing methods of label swapping in the transit nodes (BGP-LU, RSVP, LDP)
- Yet do NOT create a full mesh of signaled LSPs between all ToRs (N$^2$)

# SPRING Use-Case #3: Egress Peer Engineering

## NOT a New Idea in This Community

NANOG48
*February 2010*

"BGP-TE: Combining BGP and MPLS-TE to Avoid Congestion to Peers"

BGP Traffic Engineering Using RSVP-TE?

Tom Scholl
AT&T Labs
<tom.scholl@att.com>

Richard Steenbergen
nLayer Communications
<ras@nlayer.net>

- The concepts
  - Create an overlay that terminates at the peering router
    - It may start at the source host, or at the ingress router
  - Use this overlay to
    - Bypass the route lookup process at the peering router
    - Override the BGP best-path selection (possibly using application performance feedback)

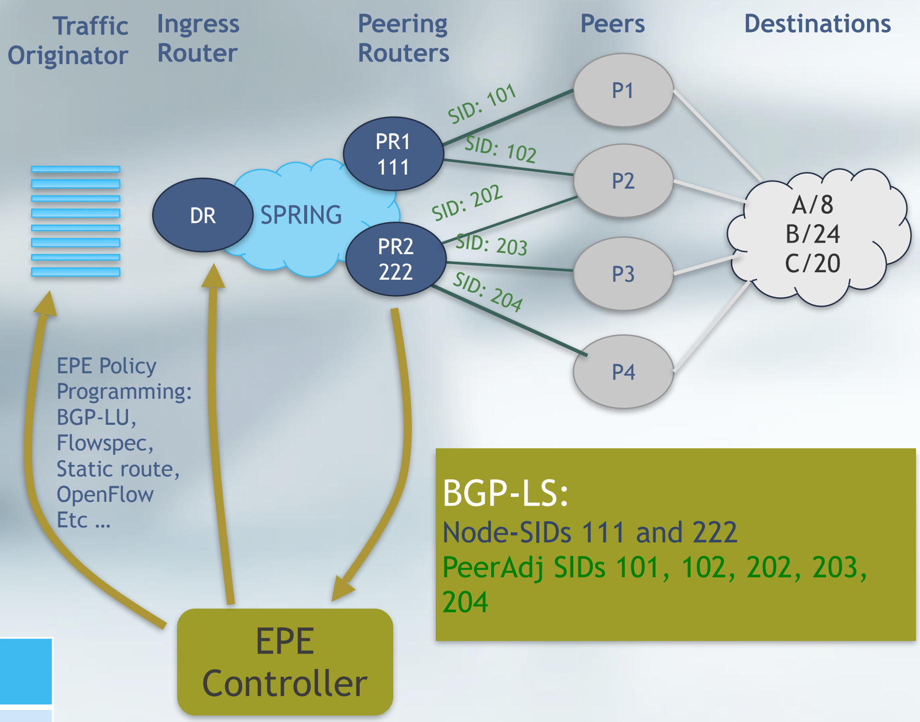
# SPRING Use-Case #3: Egress Peer Engineering

## Reference: draft-filsfils-spring-segment-routing-central-epe

- Role of PR:
  - Assign per-peer labels
  - Announce own loopback with label
  - Announce routes to controller
  - De-capsulate outbound traffic (Data-plane)
- Role of Controller
  - Make best route selection
  - Generate encapsulation for overlay
  - Program ingress with proper encapsulation
- Role of Ingress
  - Impose encapsulation on packets

Example EPE Overlay Policy

| POLICY | STACK |
|---|---|
| *For A.0/16 (first half of A/8) send to P1* | *111*<br>*101* |
| *For A.128/16 (second half of A/8), send to PR2, then P4* | *222*<br>*204* |
| *For B/24, send via PR2, then P3* | *222*<br>*203* |
| *For C/20, send to PR1, then P2* | *222*<br>*202* |



BGP-LS:
Node-SIDs 111 and 222
PeerAdj SIDs 101, 102, 202, 203, 204

| Traffic Origination | Likely Overlay Encapsulation |
|---|---|
| *Data Center* | *MPLS over GRE (or GRE-only)* |
| *CDN Cache* | *MPLS over MPLS* |

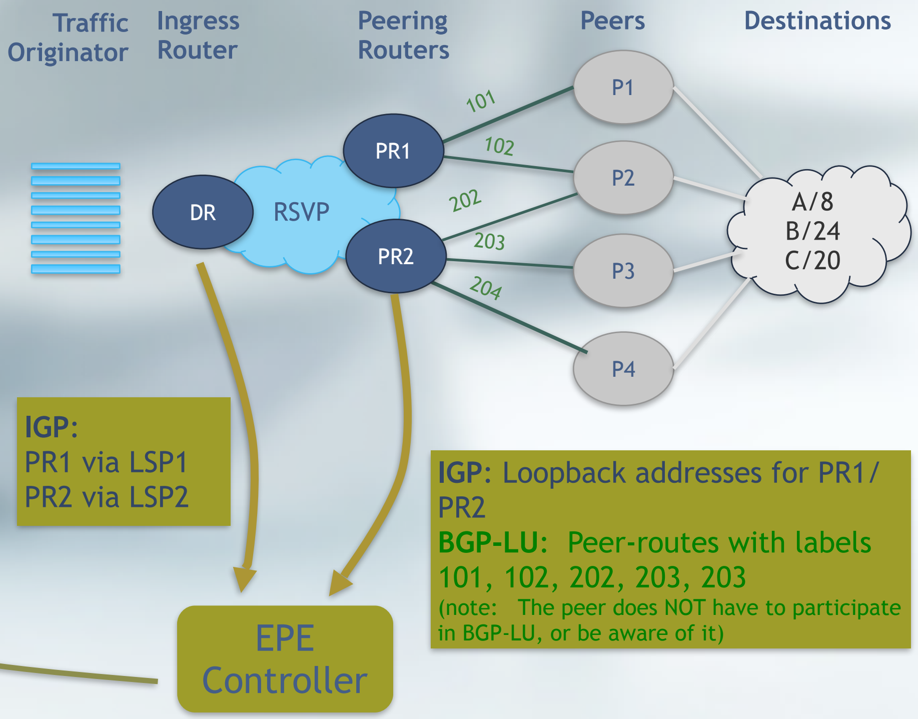
# SPRINGspiration #3: Egress Peer Engineering

## Same Use-Case, This Time Without SR, Just BGP-LU

- Standard BGP-LU
  - Used to allocate per-peer label to the /32 identifying the peer
  - The peer is unaware
- This works well with current protocols
  - Deployed extensively

Example EPE Overlay Policy

| POLICY | STACK |
|---|---|
| *For A.0/16 (first half of A/8) send to P1* | *LSP 1*<br>*101* |
| *For A.128/16 (second half of A/8), send to PR2, then P4* | *LSP 2*<br>*204* |
| *For B/24, send via PR2, then P3* | *LSP 2*<br>*203* |
| *For C/20, send to PR1, then P2* | *LSP 2*<br>*202* |



**IGP:**
PR1 via LSP1
PR2 via LSP2

**IGP:** Loopback addresses for PR1/PR2
**BGP-LU:** Peer-routes with labels 101, 102, 202, 203, 203
(note: The peer does NOT have to participate in BGP-LU, or be aware of it)

Same EPE policy can be constructed

**Reference: draft-gredler-idr-bgplu-epe**

# SPRINGspiration #3: EPE

## BGP-LU Enhancements for EPE

## Auto-generation of BGP-LU routes for peers

- Based on existing EBGP session to peer.
- Instead of defining a static route and then exporting it to IBGP-LU (previous technique).
- Semantics: POP, forward to peer interface.
- Export to IBGP-LU with next-hop self
- Attach BGP communities to inform ingress / controller about the nature of the label
  - Single-hop EBGP session
  - Multi-hop EBGP session
  - Parallel multi-hop EBGP sessions to be load-balanced

## Local protection for labeled traffic

- Because we don’t want to wait for the controller to re-progam all hosts/ingress routers
- 3 protection options
  - Ordered list of backup peers
  - Remote next-host (resolved via inet[6].3)
  - IP lookup

```
# show protocols bgp
egress-te-backup-paths {
   template abc {
      peer 19.2.0.2;
      ip-forward;
   }
   template abcv6 {
      peer 19:2::2;
      peer 19:1::1;
      remote-nexthop {
         ::ffff:9.9.9.9;
      }
   }
   template def {
      peer 19.1.0.1;
      remote-nexthop {
         7.7.7.7;
      }
   }
}
group toPeer1Link1 {
   egress-te; …
}
group toPeer3V6 {
   egress-te {
      backup-path abcv6;
   } …
}
group toPeer2 {
   egress-te {
      backup-path def;
   } …
}
```

# Conclusions

## SPRING has sparked the imagination

- Around useful source/static routing.
- SPRING brings net-new use-cases and benefits but requires an infrastructure upgrade.
  - New forwarding mechanism - training, operationalizing, de-bugging, and not the least, accepting the loss of some useful features.
- By applying some of the SPRING concepts in existing networks, creative operators have achieved some of the cool-ness of SPRING and source-routing on their existing MPLS networks

## 3 Examples in this talk

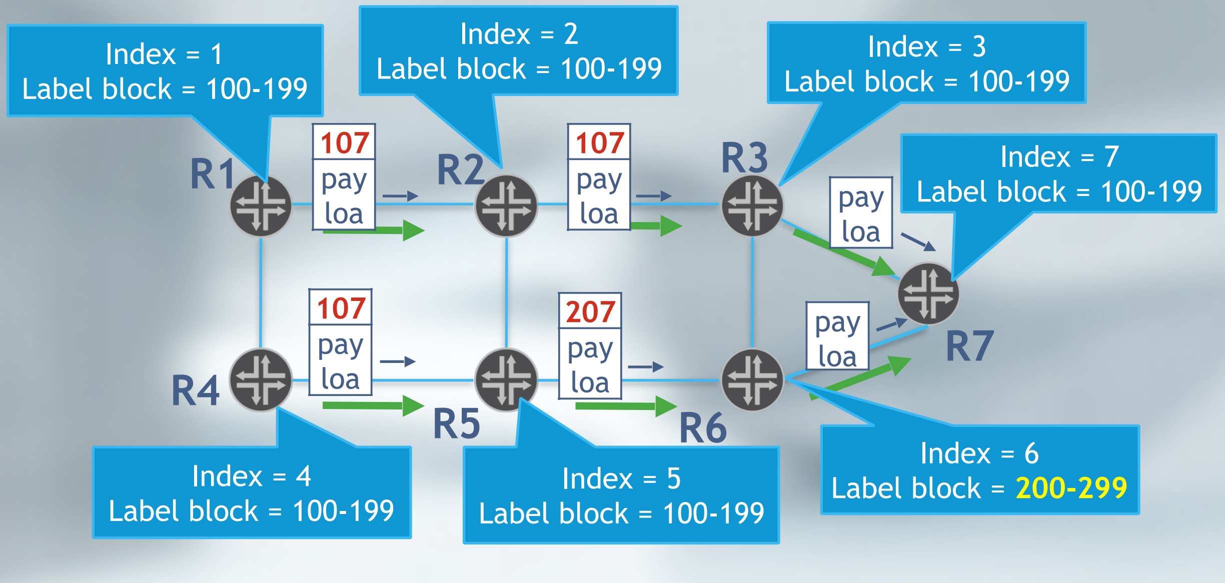
- Exhaustive network monitoring
  - Use static LSP constructs the same way adjacency labels are used to source-route OAM probes through every path in your network.
- Static LSPs with remote next-hop resolution and stitching
  - Achieve predictable “global” label assignments in the data-center using traditional MPLS transport without creating full LSP mesh between all ToRs.
- Egress Peer Engineering (EPE)
  - SPRING has sparked renewed interest in this existing solution, and has given us a reason re-think it and enhance it.

# Thank You

# Backup Slide

# SPRING: Node Label (SID)

## Local Label Ranges (SRGBs) with Global Indexes



Index = 1
Label block = 100-199

Index = 2
Label block = 100-199

Index = 3
Label block = 100-199

Index = 7
Label block = 100-199

Index = 4
Label block = 100-199

Index = 5
Label block = 100-199

Index = 6
Label block = 200-299

- Have your cake & eat it too
- Ensuring interoperability
  - Across vendors and implementations
  - With environments running RSVP/LDP/BGP-LU
- Still one can configure the same SRGB blocks on all devices
  - If they allow it
  - For a moral equivalent of global labels

```
R4:
packet destination = R7
index = 7, next-hop = R5
transmit_label  = (R5_label_offset + index)
                 = 100 + 7 = 107
```

```
R5:
index = receive_label - R5_label_offset = 107 - 100 = 7 (R7)
next-hop = R6
transmit_label = (R6_label_offset + index) = 200 + 7 = 207
```