Bandwidth and Path

Be flexible…

Peter Lundqvist
peter@arista.com

Session Agenda

"Even with the best routing design and ECMP based topology, traffic and paths do not always reflect the needs. The demand to be adaptive and be flexible regards paths usually arrives not long after any new shiny implementation. In the early days much hope was for RSVP and state machine, additions of routing to the IPv6 header but most of all MPLS. This session focus on experiences from working with innovations like Segment-routing, BGP Labeled Unicast and other ways achieve path changes based on statistical inputs from both the network and from the TCP/IP stack"

- The focus on this session will not be yet another useless vendor created scenario to show the need for traffic-engineering using RSVP-TE in a triangle scenario, or the opposite, bashing MPLS just because not understanding it…
- I will talk about based on statistics change path towards NEXT_HOP both in a static way, but also in dynamical ways of doing it…
- Basically the IP stack performance with TCP drops/retransmissions very much dependent upon the path…

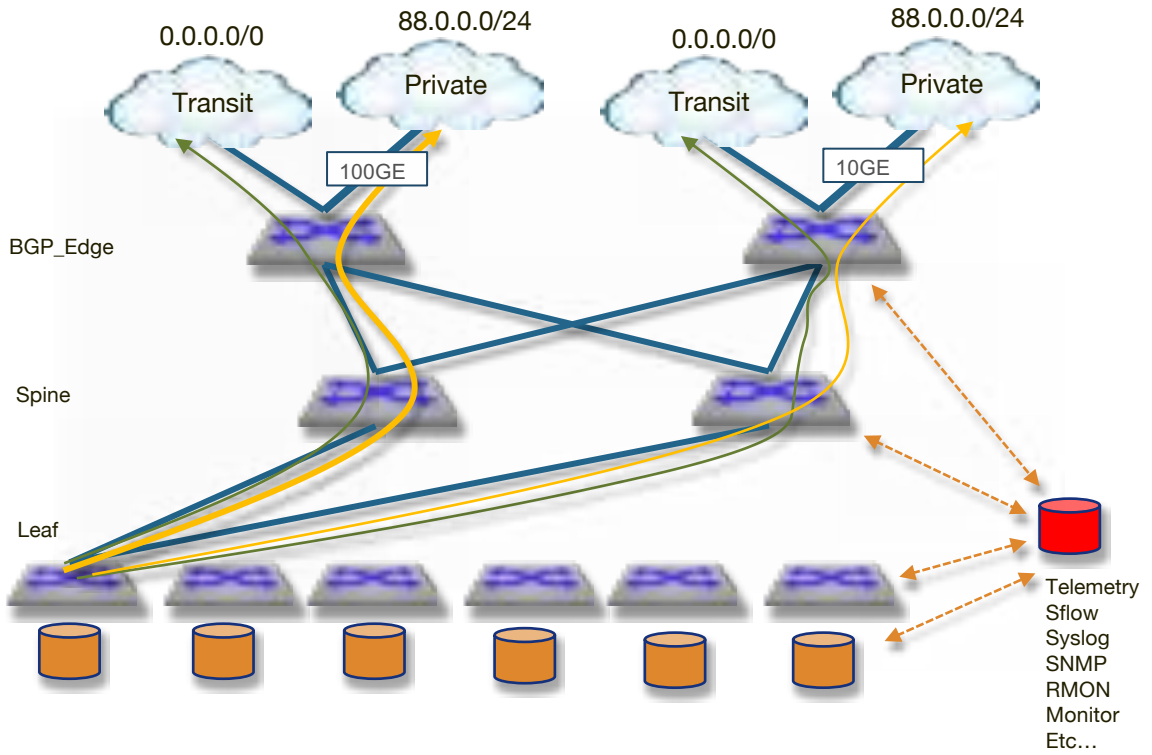
Performance = Bandwidth&Path

The bigger picture

The baseline of Performance

In Datacenter, or any network most all layers can deliver statistics

1. Sflow deliver traffic statistic
2. Telemetry deliver control plane information from all network elements.
3. Statistics from servers includes baseline regards of TCP statistics
4. Thresholds generate Logs and Alarms



Tracking the traffic matrix

- **Example from switch with Sflow extensions, the traffic matrix can be visualized**

```
petrus:~ plundqvi$ sudo nfdump -R ./nfdump/ -n 7 -s record/bytes
Aggregated flows 175
Top 7 flows ordered by bytes:
Date first seen          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port   Packets    Bytes Flows
2018-02-18 23:17:40.380  1271.332 UDP       194.22.194.2:43422 ->     239.16.16.21:5555    417900  570.9 M  4179
2018-02-19 03:20:21.546    17.911 TCP      193.10.252.19:22    ->     192.168.0.25:56332   363700  546.1 M  3637
2018-02-19 03:03:12.848  1073.610 UDP       194.22.194.6:54916 ->    239.195.0.110:5555    357600  488.5 M  3576
2018-02-18 23:17:42.380  1149.301 TCP     64.233.162.108:993   ->     192.168.0.25:51392    70400  102.6 M   704
2018-02-19 03:18:14.444    70.058 TCP      193.10.252.19:22    ->     192.168.0.25:56329   293700   22.3 M  2937
2018-02-19 03:10:24.658    73.190 TCP      193.10.252.19:22    ->     192.168.0.25:56308   287800   21.8 M  2878
2018-02-18 23:26:24.004    19.935 TCP     185.42.136.139:80    ->     192.168.0.25:53891    19400   19.1 M   194

Summary: total flows: 26439, total bytes: 2704049600, total packets: 2643900, avg bps: 1481054, avg pps: 181, avg
bpp: 1022
Time window: 2018-02-18 23:17:40 - 2018-02-19 03:21:06
Total flows processed: 26439, Blocks skipped: 0, Bytes read: 1699176
Sys: 0.012s flows/second: 2183237.0  Wall: 0.010s flows/second: 2590788.8
```

Tracking the traffic matrix…

- **Example break down to active prefixes in the network**

```
petrus:~ plundqvi$ sudo nfdump -R ./nfdump/ -A srcip4/24 -s record/bytes
Aggregated flows 41
Top 10 flows ordered by bytes:
Date first seen          Duration       Src IP Addr   Packets     Bytes      bps     Bpp Flows
2018-02-18 23:17:40.380 14606.078      194.22.194.0    830000    1.1 G   591801    1301  8300
2018-02-18 23:21:39.331   305.556      185.42.136.0    717300  826.3 M   21.6 M    1152  7173
2018-02-19 03:06:09.958   869.499      193.10.252.0    980800  644.0 M    5.9 M     656  9808
2018-02-18 23:17:42.380  1234.022      64.233.162.0     70700  102.8 M   666480    1454   707
2018-02-18 23:17:57.949     4.266      194.132.85.0     15200   22.4 M   42.0 M    1472   152
2018-02-18 23:17:52.580     5.080      213.132.98.0     14000   20.7 M   32.7 M    1481   140
2018-02-18 23:17:51.380   233.210        23.53.36.0      1100    1.4 M    47524    1259    11
2018-02-18 23:17:51.819 14406.667       192.168.0.0      3200    1.2 M      670     377    32
2018-02-18 23:31:23.471 13325.320         23.14.4.0       600   768400      461    1280     6
2018-02-18 23:21:34.612   284.403        5.178.76.0       300   456600    12843    1522     3

Summary: total flows: 26439, total bytes: 2704049600, total packets: 2643900, avg bps: 1481054, avg pps: 181, avg
bpp: 1022
Time window: 2018-02-18 23:17:40 - 2018-02-19 03:21:06
Total flows processed: 26439, Blocks skipped: 0, Bytes read: 1699176
Sys: 0.010s flows/second: 2478811.2  Wall: 0.008s flows/second: 3285164.0
```

Where are the possible bottlenecks ?

Leaf<>Spine ?

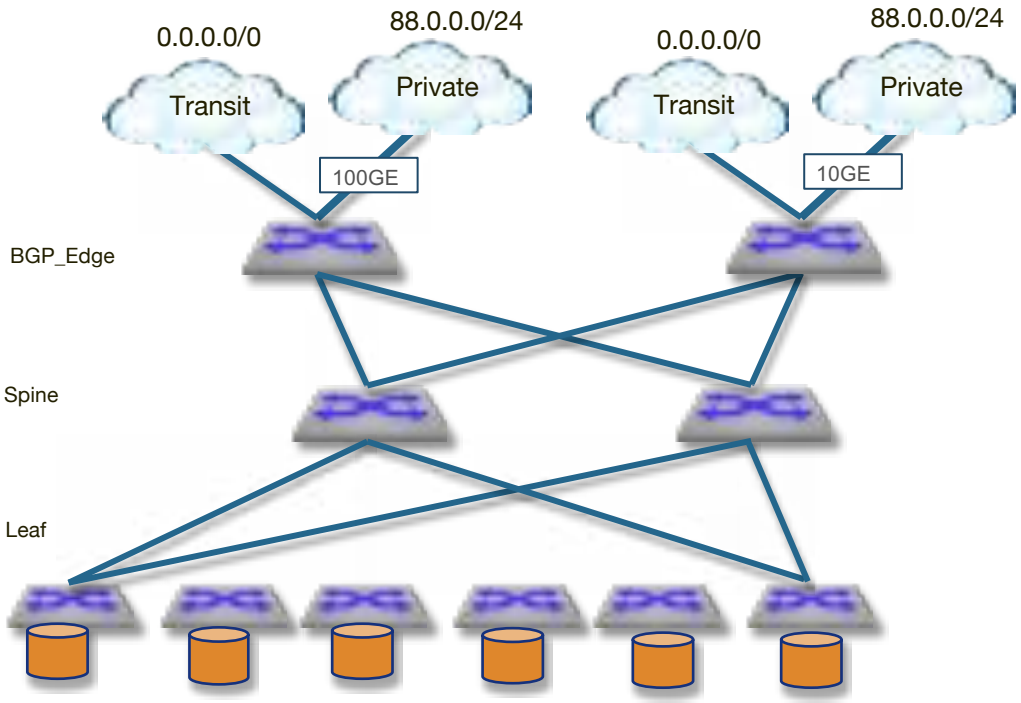
- TCP Anycast & Microburst
- Queue stats, thresholds
- Answer: bandwidth and buffers (since its simple to add)

BGP Edge

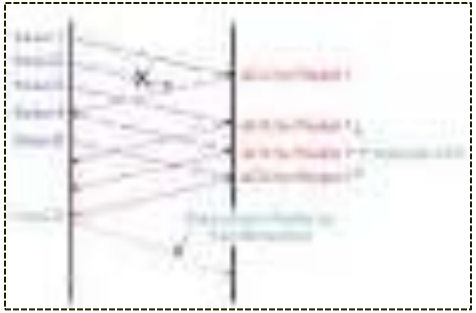
- External peering not trivial
- AS-PATH vs Quality
- Answer: monitoring and choosing the right transit for the right traffic

Server

- Stack statistics
- CPU, disk, memory,
- VM vs baremetal design
- Answer: Logs and alarms



Datacenter,
where the quality is measured from the servers



Statistics report utilization issues: the ratio of segment retransmission increase

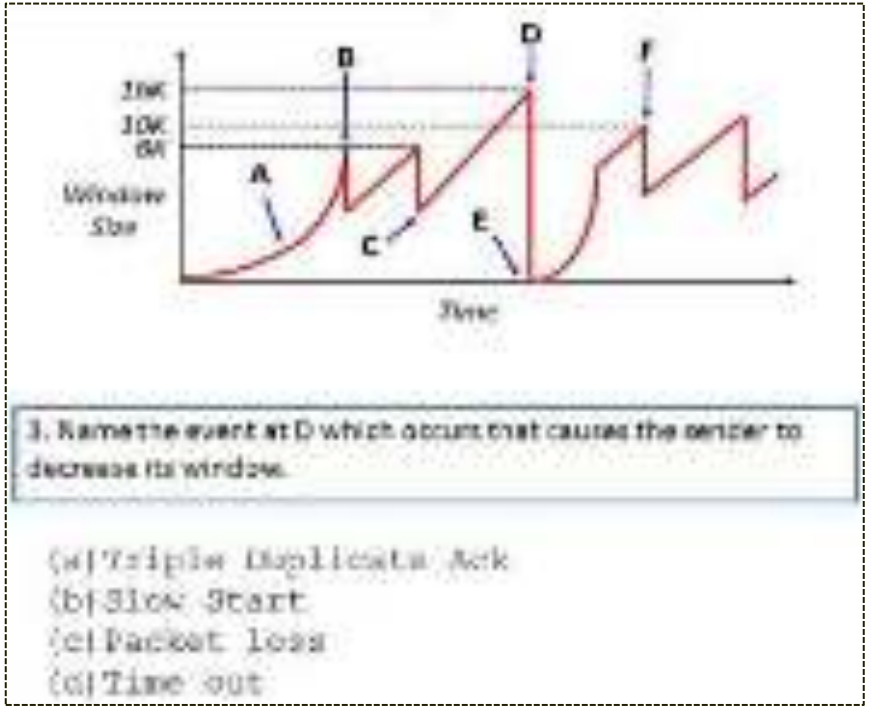
What can be the issue?

1. ECMP or links East-West traffic (Leaf-Spine)
2. BGP Edge Peering
3. DDoS
4. “Svensk Ridsport” vote flood for the Jerringpriset ?

```
sysadmin@S09SFN01:~$ netstat -s | grep -E 'send|retransmited'
    917882534 segments send out
    3706544 segments retransmited

sysadmin@S09SFN02:~$ netstat -s | grep -E 'send|retransmited'
    3331949939 segments send out
    3952631 segments retransmited

sysadmin@S09SFN03:~$ netstat -s | grep -E 'send|retransmited'
    3810795373 segments send out
    3656219 segments retransmited

sysadmin@S09SFN04:~$  netstat -s | grep -E 'send|retransmited'
    3619684592 segments send out
    3208939 segments retransmited
(…)
```

Several TCP stacks out there…

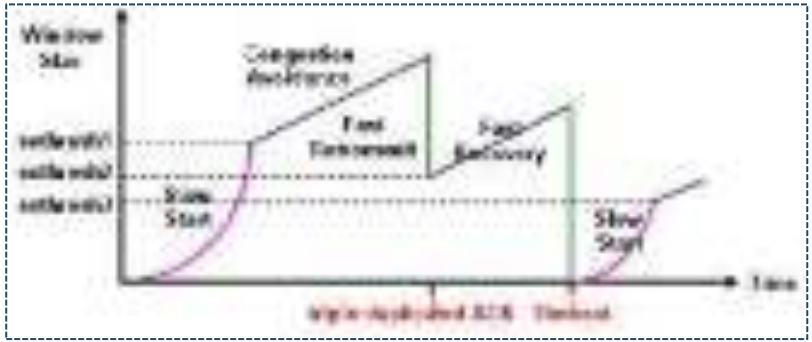
- **Reno, New Reno, Vegas, Cubic…**
  - Wait for RTO vs react on Duplication ACK
  - Fast retransmission, Fast recovery, SACK…
  - Cwnd, ssthresh etc…
  - Bla…
- **However this session is not a lecture regards TCP congestion avoidance ☺**



Tracking…

Digging deeper

- Much data can be taken by analyze captures from problematic sessions
- This can automatically be started based on increased DupACK/ RTO seen earlier

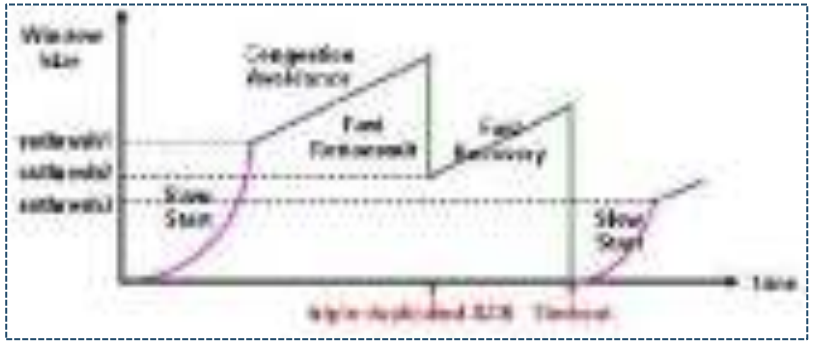



```
petrus:~ plundqvi$ tcptrace -lr trouble.pcap
(…)
TCP connection 1:
     host a:        petrus.lan:56286
     host b:        server.from.hell:pcanywherestat
     complete conn: yes
     first packet:  Sat Feb 17 23:29:49.325240 2018
     last packet:   Sat Feb 17 23:30:00.645757 2018
     elapsed time:  0:00:11.320517
     total packets: 13811
(..)
   a->b:                                b->a:
     total packets:          6441         total packets:          7370
(..)
     RTT samples:              96         RTT samples:            3161
     RTT min:                14.6 ms      RTT min:                 0.0 ms
     RTT max:               209.7 ms      RTT max:                23.6 ms
     RTT avg:                23.0 ms      RTT avg:                 0.2 ms
     RTT stdev:              27.7 ms      RTT stdev:               0.8 ms
     RTT from 3WHS:          15.9 ms      RTT from 3WHS:           0.1 ms
     RTT full_sz smpls:          2        RTT full_sz smpls:         2
     RTT full_sz min:        15.9 ms      RTT full_sz min:         0.1 ms
     RTT full_sz max:        16.3 ms      RTT full_sz max:         0.1 ms
     RTT full_sz avg:        16.1 ms      RTT full_sz avg:         0.1 ms
     RTT full_sz stdev:       0.0 ms      RTT full_sz stdev:       0.0 ms

     post-loss acks:             0        post-loss acks:         1130
     segs cum acked:             1        segs cum acked:         2973
     duplicate acks:             1        duplicate acks:          670
     triple dupacks:             0        triple dupacks:            2
     max # retrans:              0        max # retrans:             0
     min retr time:           0.0 ms      min retr time:           0.0 ms
     max retr time:           0.0 ms      max retr time:           0.0 ms
     avg retr time:           0.0 ms      avg retr time:           0.0 ms
     sdv retr time:           0.0 ms      sdv retr time:           0.0 ms
```

Tracking…

Analyze outcome

- Important to analyze outcome, not just “ok it works, lunch”
- Here the action taken was to change the BGP Edge Peering transit and the path




```
petrus:~ plundqvi$ tcptrace -lr trouble.pcap
(…)
TCP connection 1:
     host a:        petrus.lan:56286
     host b:        server.from.hell:pcanywherestat
     complete conn: yes
     first packet:  Mon Feb 19 03:06:05.639846 2018
     last packet:   Mon Feb 19 03:06:10.408576 2018
     elapsed time:  0:00:04.768729
     total packets: 9112
(…)
   a->b:                                b->a:
     total packets:          1757         total packets:          7355
(…)
     RTT samples:              98         RTT samples:            1472
     RTT min:                2.1 ms       RTT min:                0.0 ms
     RTT max:              201.8 ms       RTT max:                0.3 ms
     RTT avg:                8.8 ms       RTT avg:                0.1 ms
     RTT stdev:             27.8 ms       RTT stdev:              0.0 ms
     RTT from 3WHS:          2.6 ms       RTT from 3WHS:          0.0 ms
     RTT full_sz smpls:         2         RTT full_sz smpls:         2
     RTT full_sz min:        2.4 ms       RTT full_sz min:        0.0 ms
     RTT full_sz max:        2.6 ms       RTT full_sz max:        0.1 ms
     RTT full_sz avg:        2.5 ms       RTT full_sz avg:        0.1 ms
     RTT full_sz stdev:      0.0 ms       RTT full_sz stdev:      0.0 ms

     post-loss acks:            0         post-loss acks:            3
     segs cum acked:            1         segs cum acked:         5856
     duplicate acks:            1         duplicate acks:           23
     triple dupacks:            0         triple dupacks:            0
     max # retrans:             0         max # retrans:             0
     min retr time:          0.0 ms       min retr time:          0.0 ms
     max retr time:          0.0 ms       max retr time:          0.0 ms
     avg retr time:          0.0 ms       avg retr time:          0.0 ms
     sdv retr time:          0.0 ms       sdv retr time:          0.0 ms
```

Bandwidth control

Egress Peering Engineering (EPE) **Bottleneck**

ARISTA

I know what you will say !

Come one…build the network right from start, lets go to the Bar !

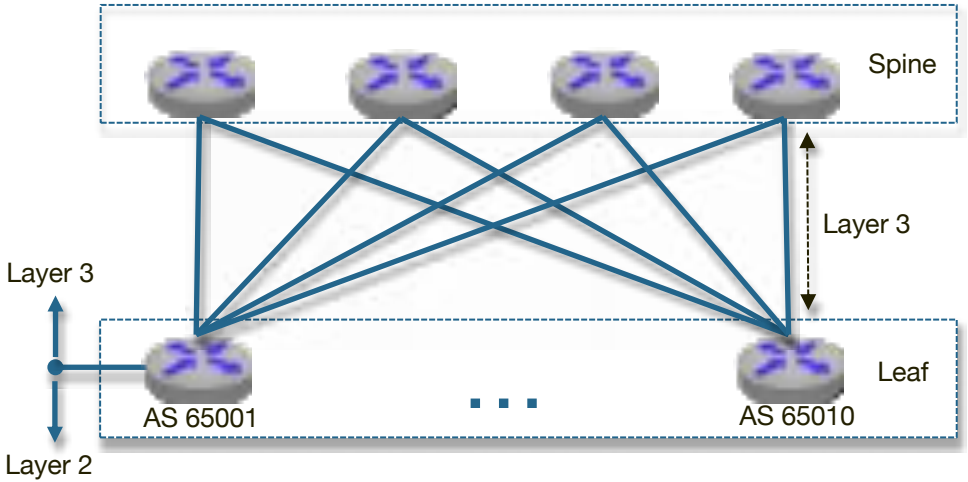
- Leaf-Spine Topology
  - ECMP and deterministic delay
- Be aware of overprovisioning
  - 48*10GE Access and 6*100GE Uplinks = No overprovisioning
  - 48*10GE Access and 4*40GE Uplinks = Overprovisioning
- Buffers
  - Avoids microbursts (interpacket Gap 1GE 96ns,10GE 9,6ns, 100GE 0.96ns
  - TCP incast (suddenly everyone want to talk to the same VM/Host)
- **Lets do that !**

Datacenter IP Fabric Design Principle #1

Stability and Scale

- If you build the largest DCs fabric in the world, you need to be able to scale the Leaf-Spine architecture as you grow
- Optimized for East-West
- Always use standard mature control protocols between the leaf and spine
  - Routing protocol between leaf and spine node
- Leaf nodes running Default Gateway for the hosts/subnets within the rack
  - Stability by reducing scope of the Layer 2 broadcast domains and MAC table size

Trust the fact that if a L2 fabric would scale, DCs would be build in that way..

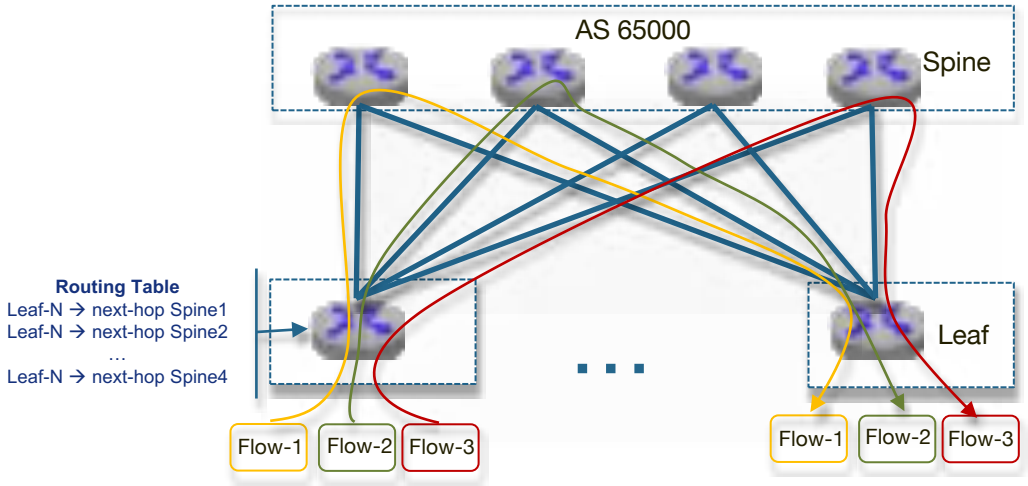


L3 Equal Cost “Multipath” Principle #2

Its not that difficult actually

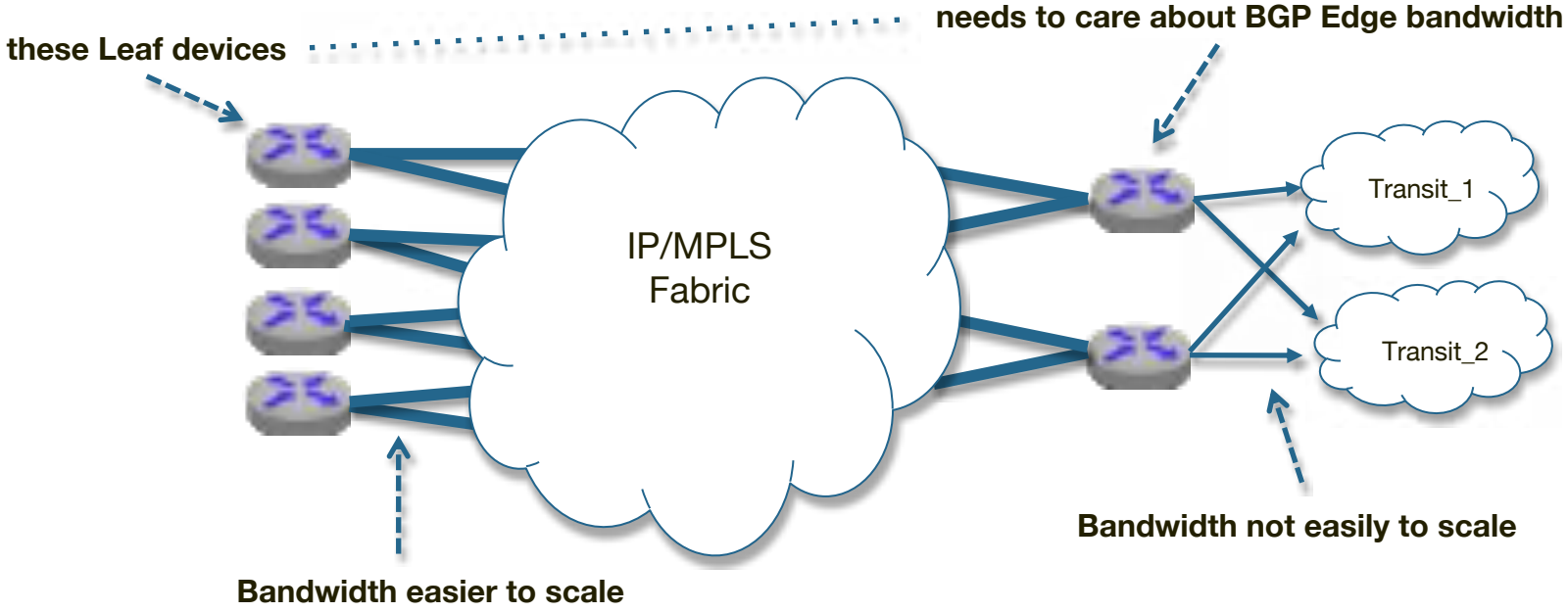
- Each leaf node has multiple paths of equal “cost” to each individual spine
- ECMP load balance flows across the multiple spine nodes, loss of a single spine means less bandwidth but NOT connectivity
- Switch load-balancing algorithm is configurable in regards of 5-tuple plus possible seed bits to avoid polarization…

Active paths, the best redundancy there is…



Why then the need for path control in a Leaf-Spine ?

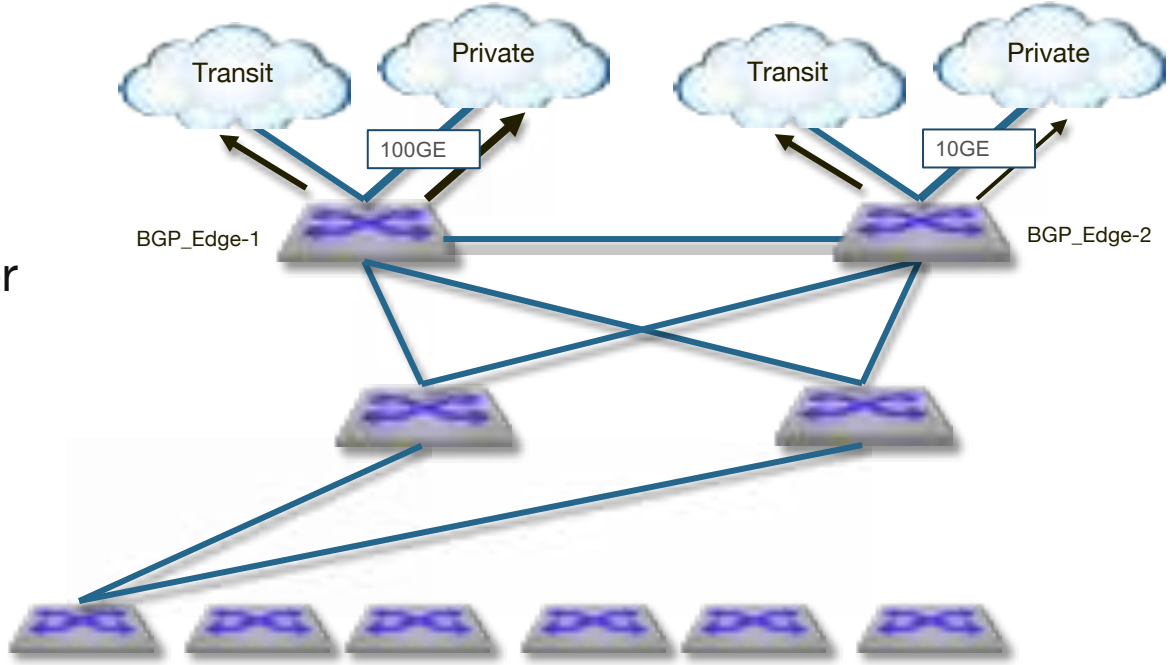
Egress Peering Engineering (EPE)



Routing handle path(s), not really bandwidth

Best path

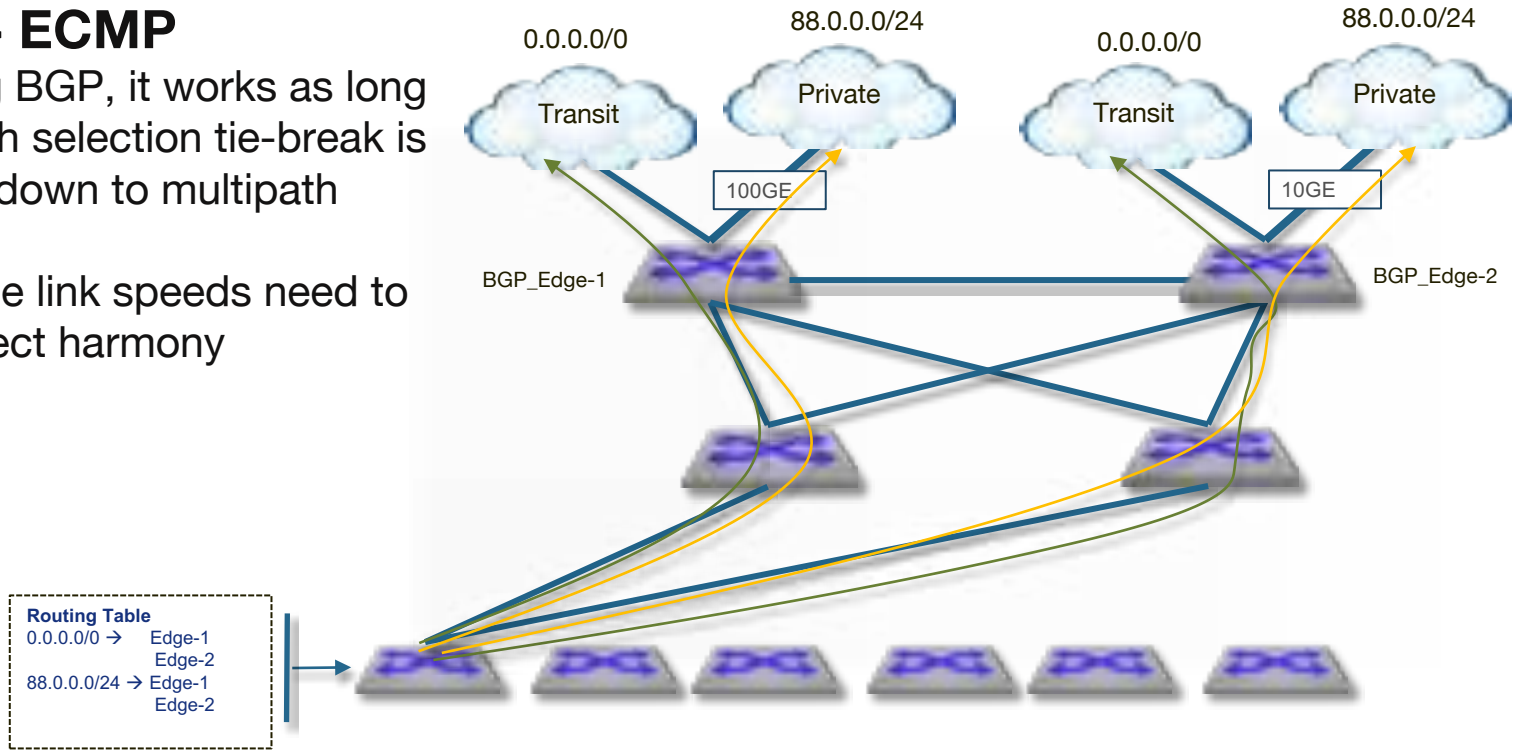
- Path selection picks a route thereby the path
- In brief its ECMP or not…
- Interface speed (or any load metric) not part of criteria for BGP.
- Even if IGP have interface metrics, still not bandwidth aware
- Traffic itself not really weighted



Equal Cost Multipath (ECMP)

Multipath => ECMP

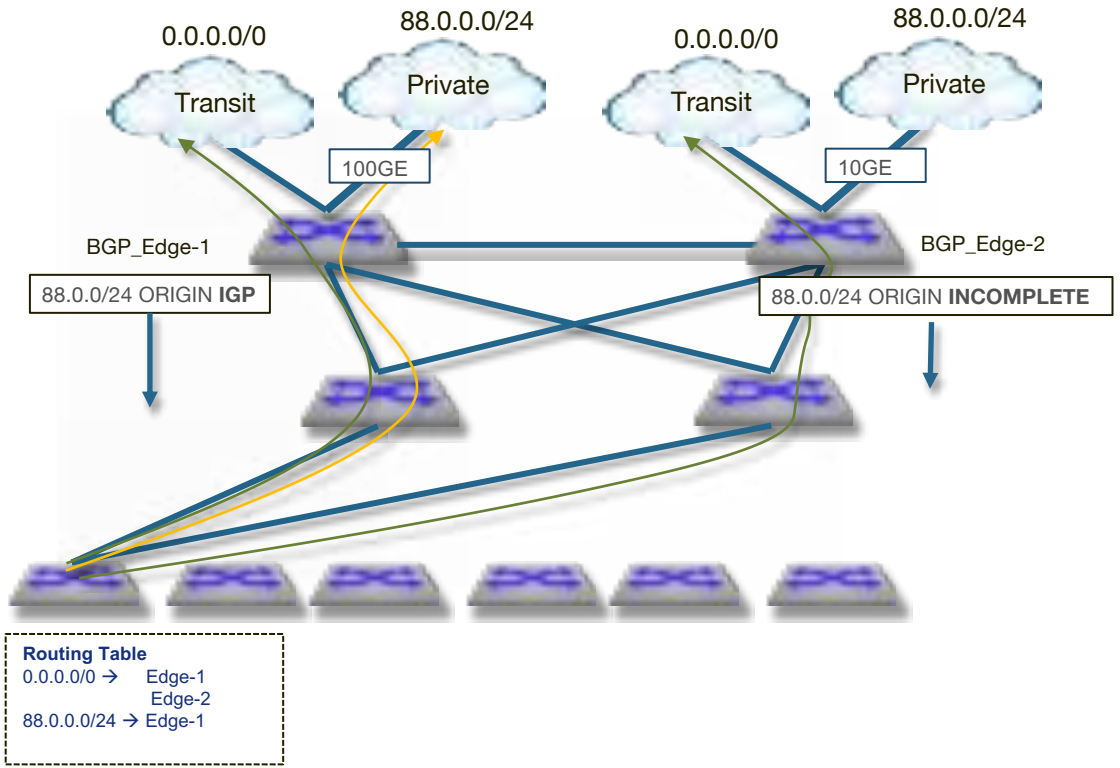
- Regarding BGP, it works as long as the path selection tie-break is the same down to multipath attribute
- For IGP the link speeds need to be in perfect harmony



ECMP with more Specific route

Of course you can use policies that affect path selection

- Multipath to some routes (thereby ECMP) example Transit
- More specific routes via private peering no multipath (thereby no ECMP) for those prefixes
- BGP Edge Policies handle the active paths

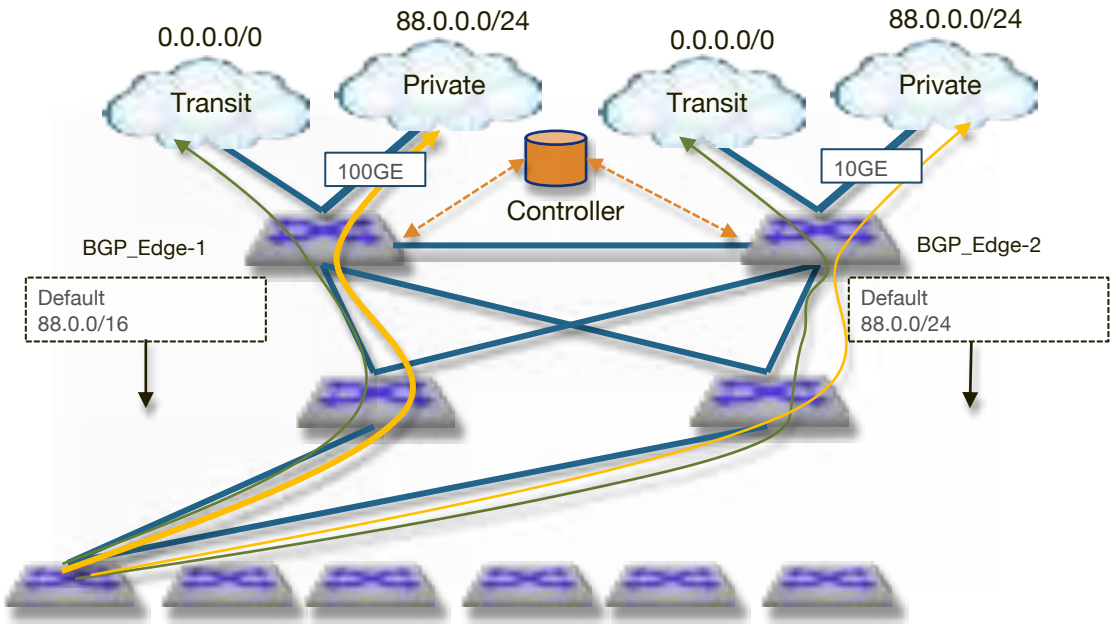


ECMP with more Specific Routes… Change paths based on collected info

Of course you can build a reactive model by monitoring statistics

On the BGP Edge:

1. Enable sflow to get statistic and/or collect interface statistics with telemetry
2. Advertise ALL routes to a controller with ADD-PATH TX
3. Change Adj-RIB-Out policy by generating more specific routes and thereby move traffic paths



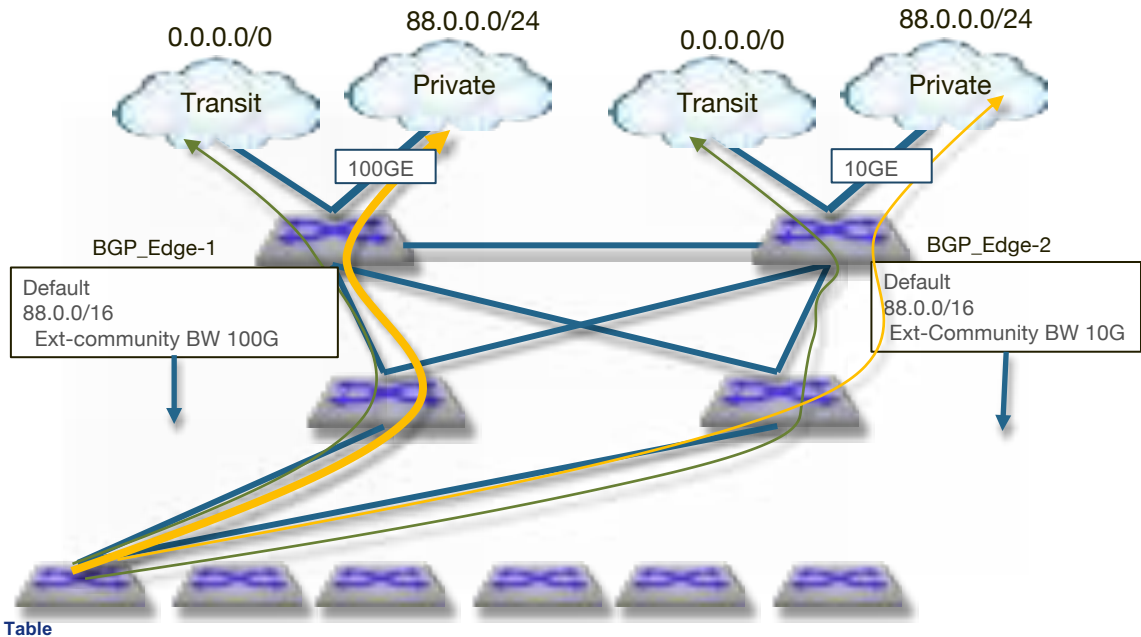
Routing Table
0.0.0.0/0 → Edge-1
Edge-2
88.0.0.0/16 → Edge-1
88.0.0.0/24 → Edge-2

ECMP but introduce weight that affect hashing

Same as the reactive model, but affect the hashing and not just using policies like a sledgehammer

On the BGP Edge:

1. Set BW towards private peers on each BGP edge reflect pipe speed
2. Advertise Route with EXTENDED_COMMUNITIES that reflect the link bandwidth



Routing Table

0.0.0.0/0 → Edge-1
Edge-2

88.0.0.0/24 → Edge-1 - 90% of Flows
Edge-2 - 10% of Flows

BGP LBW EXTENDED_COMMUNITIES

```
(…)
Border Gateway Protocol - UPDATE Message
Path Attribut - AS_PATH: 64514 64515
Path Attribut - NEXT_HOP: 42.0.0.2
(…) Path Attribut - EXTENDED_COMMUNITIES
            Flags: 0xc0: Optional, Transitive, Complete
                1... .... = Optional: Optional
                .1.. .... = Transitive: Transitive
                ..0. .... = Partial: Complete
                ...0 .... = Length: Regular length
            Type Code: EXTENDED_COMMUNITIES (16)
            Length: 8
            Carried extended communities: (1 community)
                Community: ASN 64514, 80000.000 Mbps
(…)
Network Layer Reachability Information (NLRI)
        0.0.0.0/0
            NLRI prefix length: 0
            NLRI prefix: 0.0.0.0 (0.0.0.0)
(…)
```

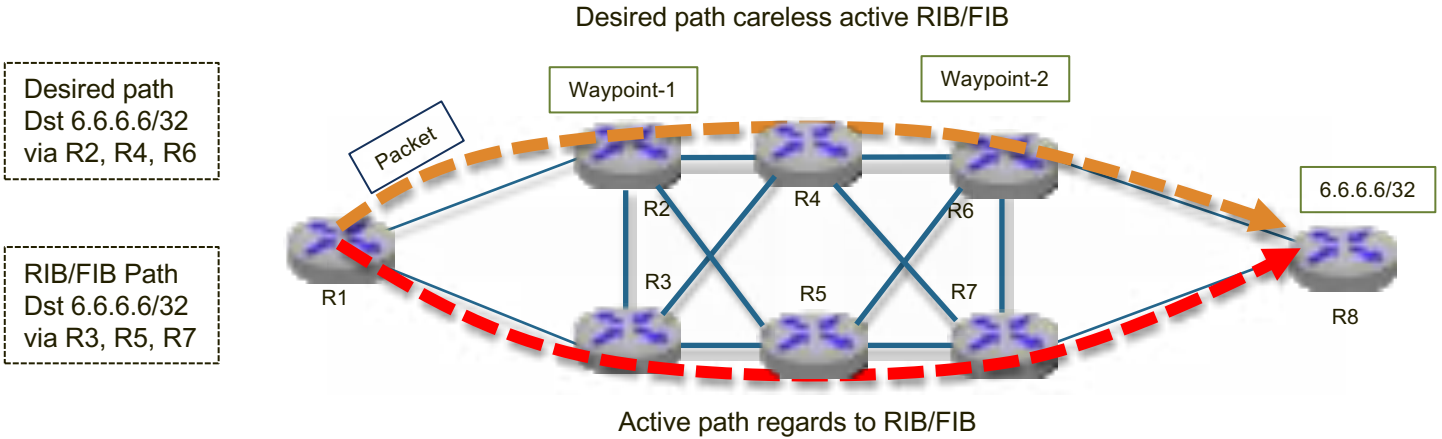
Path control

The concept of source routing

ARISTA

Source routing vs destination routing?

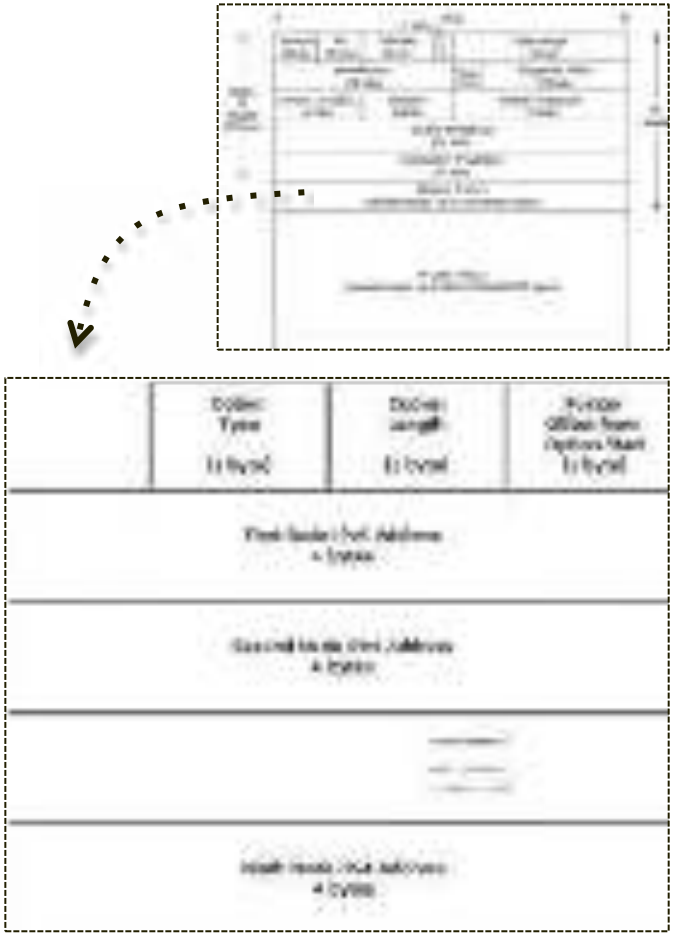
- In brief with source-routing, the source decides the path for the packet, careless of transit nodes destination based routing created by the active routing vs. forwarding table (RIB&FIB)
- Reasons for desired path can be to solve bottlenecks due to congestion, application demanding low delay etc…
- There are many ways to achieve this… some more and less useful



What is the history ?

For IPv4 its either Loose vs Strict Source Routing options (LSSR vs SSR) defined by IPv4 RFC791

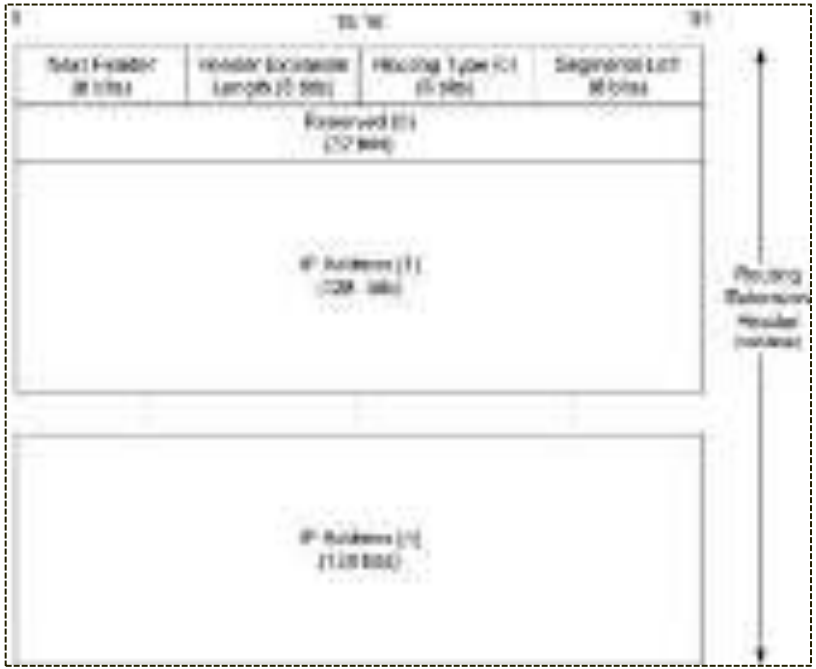
- *"The loose vs strict source and record route option provides a means for the source of an internet datagram to supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information."*
- Types
  - Type 131 (10000011) Loose Source
  - Type 137 (10001001) Strict Source
- **The tools there… anyone used it ???**



What is the history…

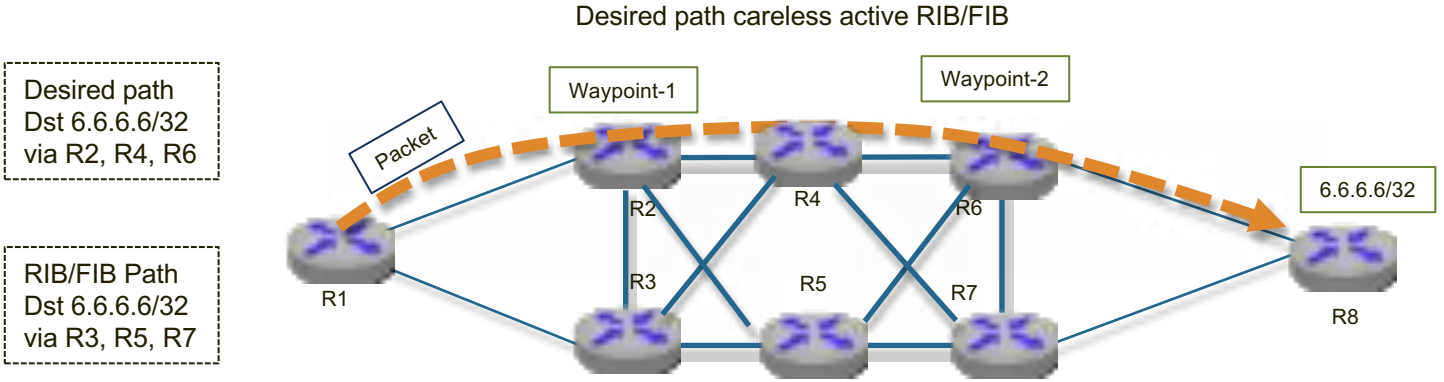
IPv6 source routing are implemented as extension header that is found after IPv6 header and before the payload itself.

- Extension headers are different than normal option headers and not in theory limited in size
- RFC 2460 defines the IPv6 Routing header Type 0 (RH0). But there are also variants Type 2 (Mobile networks) etc…
- Due to the security flaw with this approach, most TCP/IP stack implementations have RH disabled/not supported
  - RFC5095 “Deprecation of Type 0 Routing Headers in IPv6”



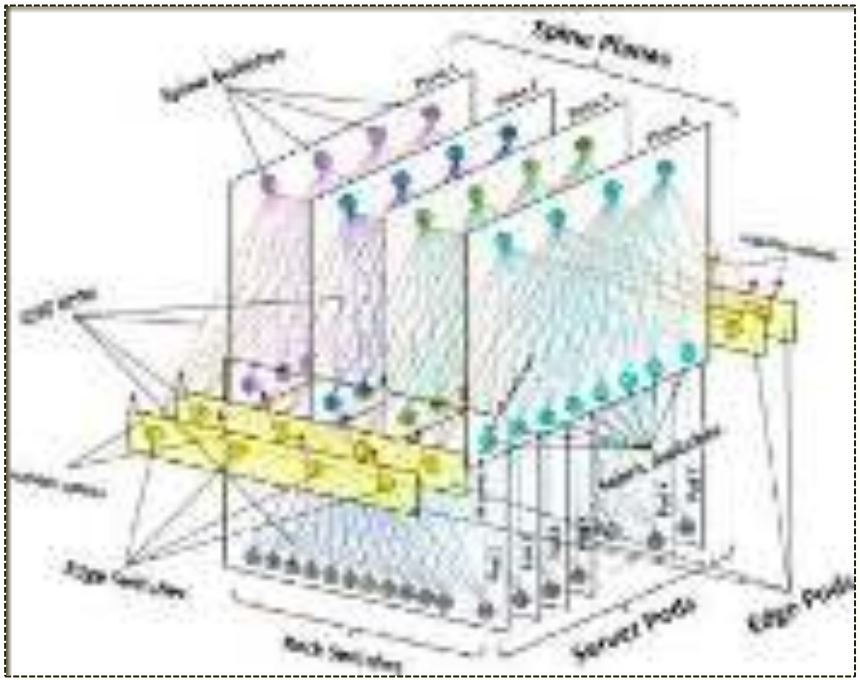
Source routing…

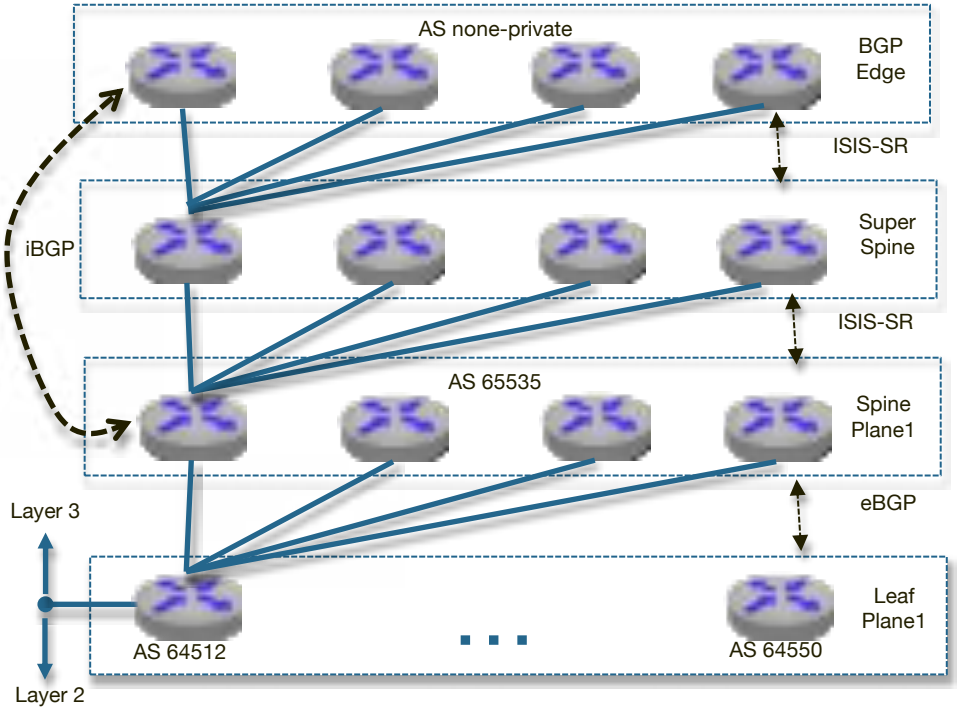
- IPv4 source routing using the concept of waypoint addresses reside in the inside option part of the earlier shown header.
  - The length of the address list can be up to 40 bytes result a maximum of 8-9 waypoints
- IPv6 extension uses similar waypoint concept, but the nextheader ability just take possible payload space to be able get more room when needed.
  - The number of waypoint addresses in IPv6 extension header in theory only limited by the MTU
- **However this is a dead-end**



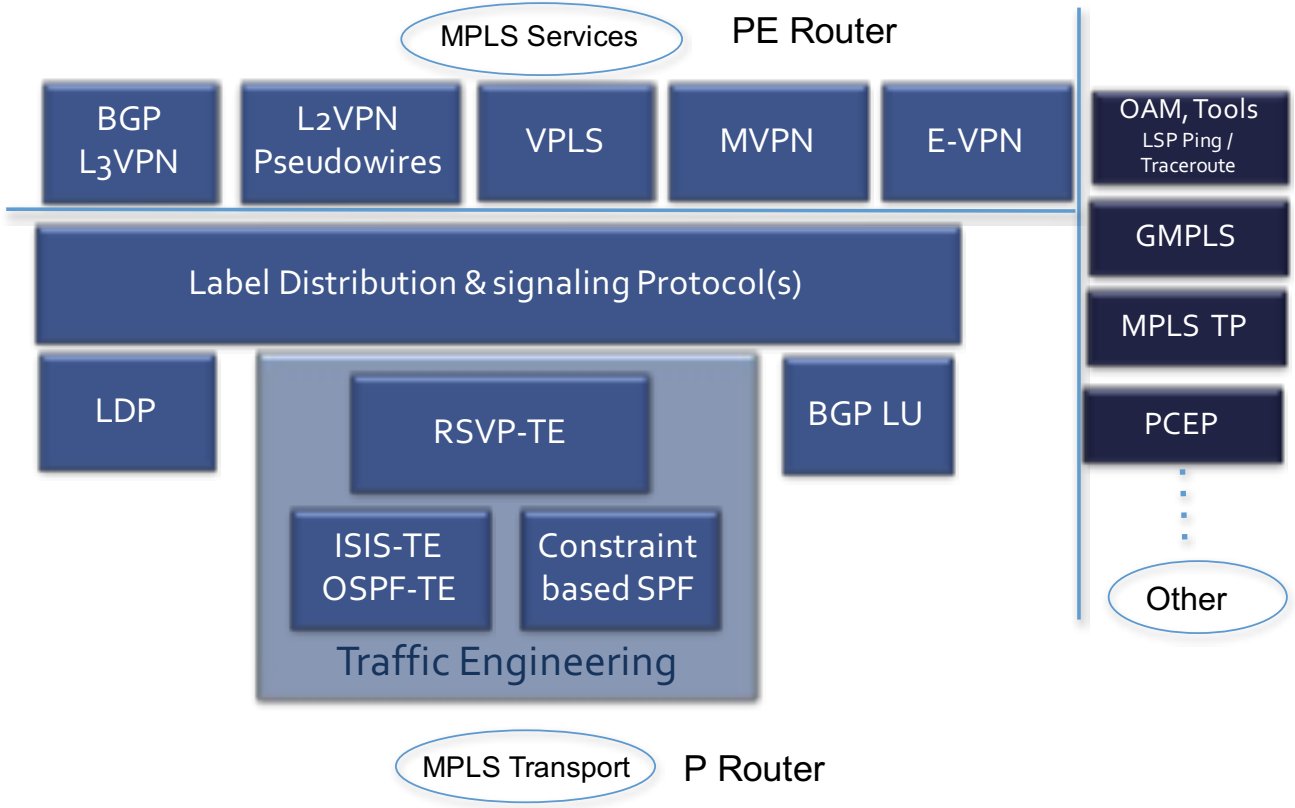
Example traffic-engineering in a Datacenter

- Paths and traffic need to be adaptable to changes and load



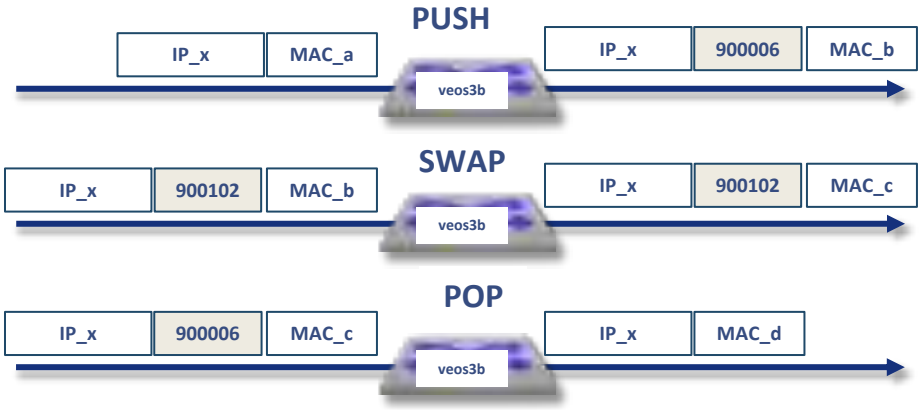


However there is a Protocol suite out there



The MPLS label Basics

- Possible actions
  - PUSH = Add Label
  - SWAP = Change label
  - POP = Remove label




```
veos3b##sh mpls ro
(…)
 900006  A[1]
                via M, 1.1.1.30, pop
                    EgressAcl: Apply
                    directly connected, Ethernet3
                    08:00:27:5a:60:e1,
 900007  A[1]
                via M, 1.1.1.41, pop
                    EgressAcl: Apply
                    directly connected, Ethernet4
                    08:00:27:9e:bb:97,
 900102  A[1]
                via M, 1.1.1.30, swap 900102
                    EgressAcl: Apply
                    directly connected, Ethernet3
                    08:00:27:5a:60:e1,
(…)
```

What about mix various controlplane solutions

- Labels are in ranges
  - Example 1-15 reserved range (example Label 3 => Tunnel/MPLS Lookup, then POP

```
veos3b#sh mpls lab ranges
Start     End       Size      Usage
-------------------------------------------
0         15        16        reserved
16        99999     99984     static mpls
100000    116383    16384     ldp (dynamic)
116384    132767    16384     isis (dynamic)
132768    362143    229376    free (dynamic)
362144    899999    537856    unassigned
900000    965535    65536     isis-sr
900000    965535    65536     bgp-sr
965536    1031071   65536     srlb
1031072   1036287   5216      unassigned
1036288   1048575   12288     l2evpn
(…)

veos3b#sh mpls tunnel fib
   Tunnel Type          Index       Endpoint           Nexthop        Interface      Labels
------------------- ----------- ------------------ -------------- ----------------- ----------
   IS-IS SR IPv4       1           1.1.1.4/32         1.1.1.40       'Ethernet1'    [ 3 ]
   LDP                 1           1.1.1.5/32         1.1.1.42       'Ethernet2'    [ 3 ]
   IS-IS SR IPv4       2           1.1.1.6/32         1.1.1.40       'Ethernet1'    [ 900006 ]
   LDP                 2           1.1.1.102/32       1.1.1.42       'Ethernet2'    [ 100000 ]
```

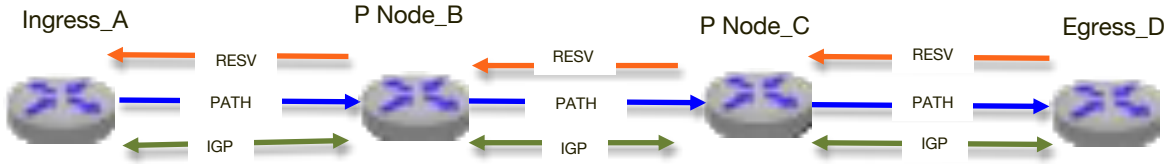
What about RSVP-TE?

Several RFC involved 3209, 4090, 4875 etc…
Doing the job, but…

- Complex, demands more or less a full-mesh
- Control plane scale has always been an issue
  - Per tunnel state at every network hop
  - State refresh/maintenance increase the state overhead
- Lack of ECMP, multiple explicit paths needed
- Difference with constraints and path computational options – vendor specific
- **Needs IGP for its compute**

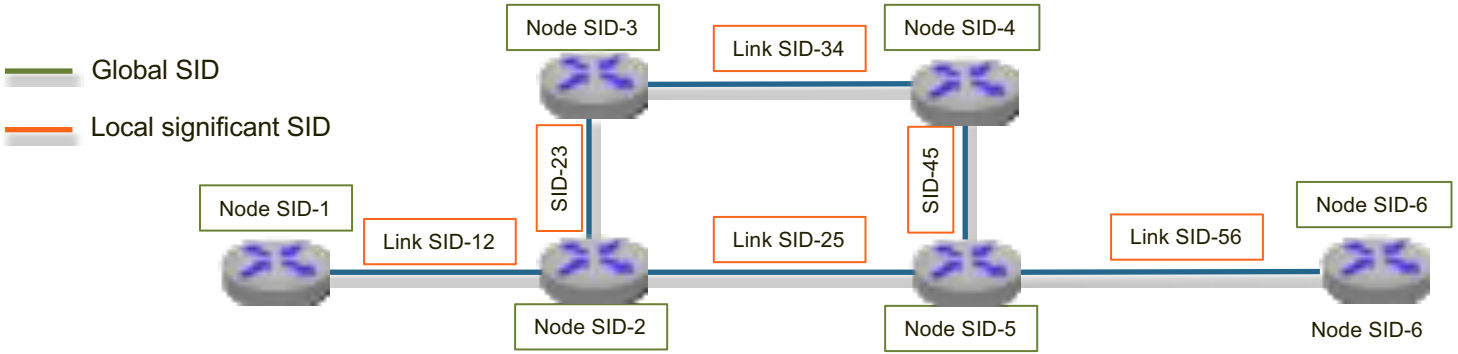
ERO
IP A>B>C>D

RECORD_ROUTE
IP B = Label 100
IP C = Label 200
IP D = Label 300



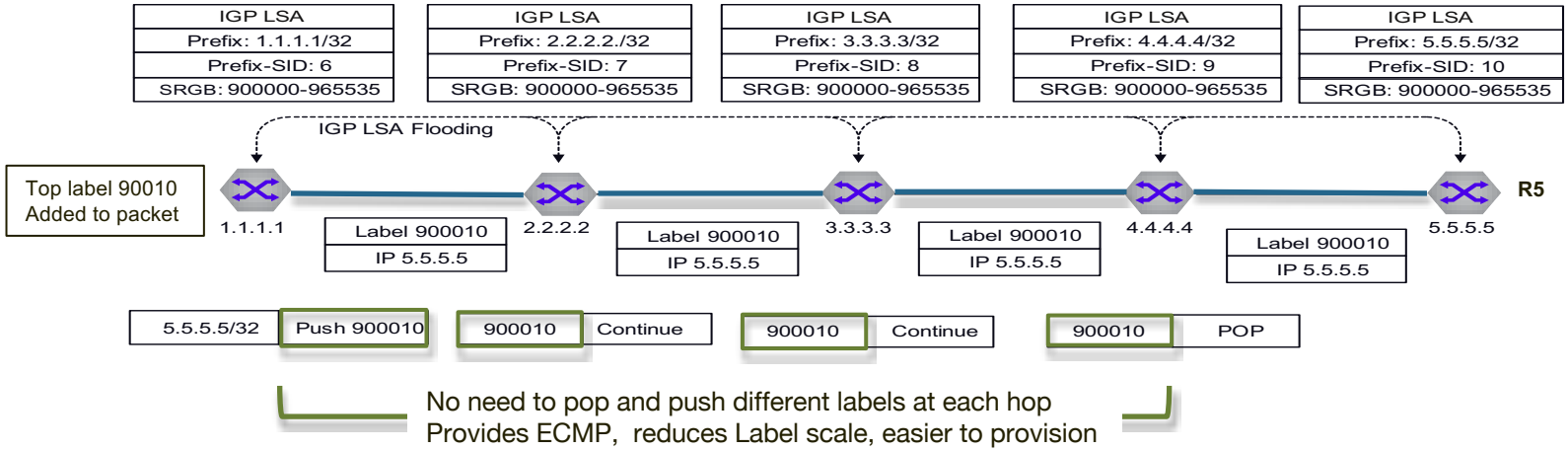
Segment Routing (SR) – Operation Overview

- SR divides the network into “segments” identified by a Segment-ID (SID)
- Global SIDs identify nodes (loopback IP), prefix or Anycast (shared IP)
  - **All nodes in the SR domain use same SID to identify the prefixes**, node or Anycast – reducing data plane state
- Local significant SIDs identify the adjacency links in the network
  - Only the originating Node understands the advertised Adjacency SID
- Both local and global SIDs are advertised as TLV extensions to the IGP (ISIS/OSPF)
- The SID is encoded as an MPLS Label in the forwarding plane



Segment Routing Operation– Global SIDs

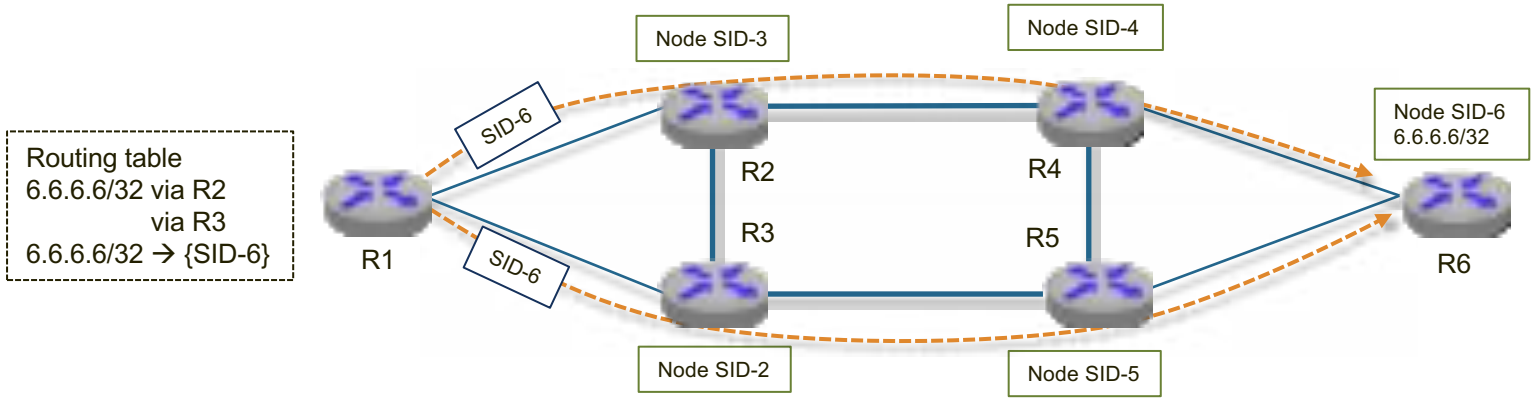
- **R5** advertises it's global node SID, which represents it's loopback IP (5.5.5.5/32)
- Advertisement via ISIS, simple sub-TLV containing prefix and corresponding SID index
- All remote nodes install the same node segment ID (90010) to 5.5.5.5 in the MPLS data plane
- At ingress, top label 900010 and to packet and reaches 5.5.5.5 via the IGP shortest path



Segment Routing – ECMP aware

The forwarding route for the global Node SID represents IGP calculated shortest path to the prefix.

- If the egress is dual-homed and supports ECMP, traffic can be load-balanced across the each of the equal cost paths
- Traffic load-balancing on 5-tuple (Prefix SID mapped to IGP ECMP SPF paths).



ISIS-SR itself do all the heavy lifting with labels

- With segment-routing there is no need for a band-aid protocol responsible for the IP<>Label mapping

```
veos3b#sh isis database detail bgp-edge2.00-00
IS-IS Instance: peter VRF: default
  IS-IS Level 2 Link State Database
    LSPID                 Seq Num   Cksum  Life  IS Flags
    bgp-edge2.00-00       6         38593  64357 L2 <>
      NLPID: 0xCC(IPv4)
      Area address: 49.0000
      Hostname: bgp-edge2
      Interface address: 1.1.100.0
      Interface address: 1.1.1.34
      Interface address: 1.1.1.102
      IS Neighbor          : bgp-edge.00         Metric: 10
        IPv4 Neighbor Address: 1.1.100.1
        IPv4 Interface Address: 1.1.100.0
        Adj-sid: 116384 flags: [ L V ] weight: 0x0
      Reachability         : 1.1.100.0/31 Metric: 10 Type: 1 Up
      Reachability         : 1.1.1.32/30 Metric: 10 Type: 1 Up
      Reachability         : 1.1.1.102/32 Metric: 10 Type: 1 Up
        SR Prefix-SID: 102 Flags: [ N ] Algorithm: 0
      Router Capabilities: 1.1.1.102 Flags: [ ]
        SR Capability: Flags: [ I ]
          SRGB Base: 900000 Range: 65536
```

```
veos3b#sh mpls segment-routing bindings
1.1.1.4/32
   Local binding:  Label: 900004
   Remote binding: Peer ID: 0010.0100.1004, Label: imp-null
1.1.1.6/32
   Local binding:  Label: 900006
   Remote binding: Peer ID: 0010.0100.1004, Label: 900006
1.1.1.7/32
   Local binding:  Label: imp-null
   Remote binding: Peer ID: 0010.0100.1004, Label: 900007
1.1.1.102/32
   Local binding:  Label: 900102
   Remote binding: Peer ID: 0010.0100.1004, Label: 900102
(...)
```

BGP labeled-unicast (LU)

- BGP Labeled Unicast (BGP-LU) defined in RFC3107
  - AFI1 SAFI4 (IPv4)
  - AFI 2 SAFI 4 (IPv6)
- In brief it means that BGP can advertise unicast route (IP Prefix) with associated MPLS label
  - BGP-LU can be used like any other BGP peering router to router
  - But can also be used between Routers and Controller as a way to program static routes with labeled paths
  - Manipulation both the BGP NEXT_HOP and physical nexthop interface

```
(…)
Border Gateway Protocol - UPDATE Message
(…)
            MP_REACH_NLRI (21 bytes)
                Flags: 0x90
                    1... .... = Optional
                    .0.. .... = Non-transitive
                    ..0. .... = Complete
                    ...1 .... = Extended length
                Type code: MP_REACH_NLRI (14)
                Length: 17 bytes
                Address family: IPv4 (1)
                Subsequent address family identifier: Labeled Unicast (4)
                Next hop network address (4 bytes)
                    Next hop: 1.1.1.40 (4)
                Subnetwork points of attachment: 0
                Network layer reachability information (8 bytes)
                    Label Stack=900006 (bottom) IPv4=3.3.3.3.2/32
(…)
```

Segment Routing Prefix SID extensions for BGP

- BGP Segment-routing (SR) goes even further with Global-Block advertisment (SRGB)
- In brief it means that you can do SR without any IGP
- Example in a BGP only Datacenter/Network
- BGP-LU the “encapsulation”

```
Border Gateway Protocol - UPDATE Message
(…)
        Path Attribute - MP_REACH_NLRI
        Address family identifier (AFI): IPv4 (1)
            Subsequent address family identifier (SAFI): Labeled Unicast (4)
            Next hop network address (4 bytes)
                Next Hop: 20.19.152.19
            Number of Subnetwork points of attachment (SNPA): 0
            Network layer reachability information (8 bytes)
                Label Stack=3 (bottom) IPv4=100.0.0.19/32
                    MP Reach NLRI Prefix length: 56
                    MP Reach NLRI Label Stack: 3 (bottom)
                    MP Reach NLRI IPv4 prefix: 100.0.0.19
        Path Attribute - BGP Prefix-SID
        Type Code: BGP Prefix-SID (40)
            Length: 21
            Label-Index: 19
                Type: 1
                Length: 7
                Reserved: 00
                Label-Index Flags: 0x0000
                Label-Index Value: 19
            Originator SRGB
                Type: 3
                Length: 8
                Originator SRGB Flags: 0x0000
                SRGB Blocks
                    SRGB Block(200000:207999)
                        SRGB Base: 200000
                        SRGB Range: 207999
(…)
```

BGP-SR do all the heavy lifting with labels

- Again, with segment-routing there is no need for a band-aid protocol responsible for the IP<>Label mapping

```
bgp-sr-19(config)#sh ip bgp labeled-unicast detail
Local MPLS label: 900311
BGP routing table entry for 100.0.0.75/32
 Paths: 1 available
  65152 65075
    20.19.152.152 labels [ 200075 ] from 20.19.152.152 (100.0.0.152)
      Origin IGP, metric -, localpref 100, weight 0, valid, external, best
      Local MPLS label: 200075, SR Label Index: 75
BGP routing table entry for 100.0.0.152/32
 Paths: 1 available
  65152
    20.19.152.152 labels [ 3 ] from 20.19.152.152 (100.0.0.152)
      Origin IGP, metric 0, localpref 100, weight 0, valid, external, best
      Local MPLS label: 200152, SR Label Index: 152
BGP routing table entry for 100.0.0.153/32
 Paths: 1 available
  65152 65153
    20.19.152.152 labels [ 200153 ] from 20.19.152.152 (100.0.0.152)
      Origin IGP, metric -, localpref 100, weight 0, valid, external, best
      Local MPLS label: 200153, SR Label Index: 153
(…)
```

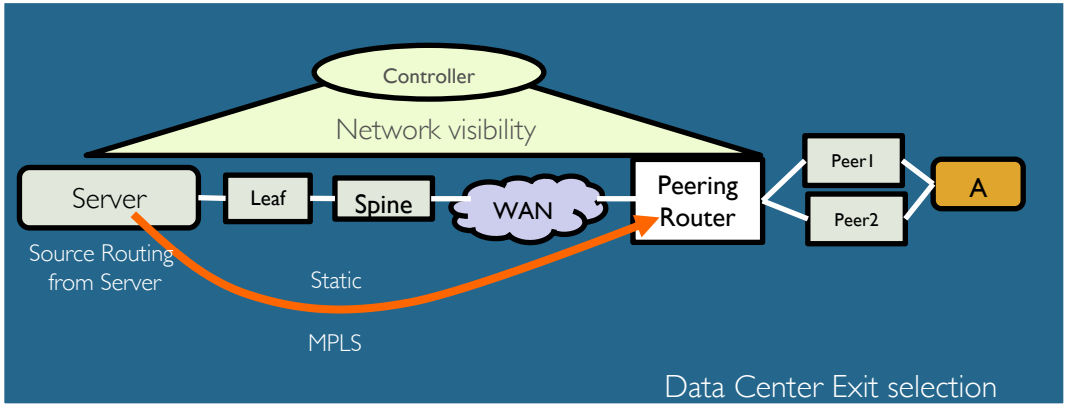
```
bgp-sr-19(config)#sh mpls lfib ro
Source Codes:
          S - Static MPLS Route, B2 - BGP L2 EVPN,
          B3 - BGP L3 VPN, R - RSVP,
          P - Pseudowire, L - LDP,
          IP - IS-IS SR Prefix Segment, IA - IS-IS SR Adjacency Segm,
          IL - IS-IS SR Segment to LDP, LI - LDP to IS-IS SR Segment,
          BL - BGP LU, ST - SR TE Policy,
          DE - Debug LFIB

 BL   200075    [1], 100.0.0.75/32
                  via M, 20.19.152.152, swap 200075
                      payload ipv4, apply egress-acl
                      interface Ethernet2
 BL   200152    [1], 100.0.0.152/32
                  via M, 20.19.152.152, pop
                      payload ipv4, apply egress-acl
                      interface Ethernet2
 BL   200153    [1], 100.0.0.153/32
                  via M, 20.19.152.152, swap 200153
                      payload ipv4, apply egress-acl
                      interface Ethernet2
```

Bandwidth & Path

Which model to choose?

Cloud/Content Providers Egress Peer Engineering

Host initiated



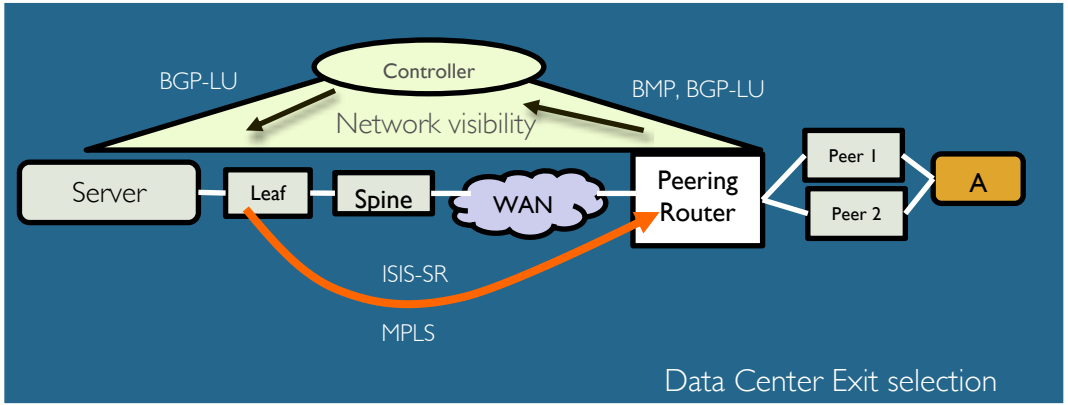
DC exit point selection or Egress Peering Engineering (EPE)

- Source routing from host/server to influence exit peering interface on Peering Router
- MPLS source routed with label swaps on P routers (static MPLS) for MPLS based L-S fabric

Controller

- Programmability and Control
- SDK/API and static MPLS
- BGP-LU can be used as Controller>Network

Cloud/Content Providers Egress Peer Engineering
Leaf Initiated



DC exit point selection or Egress Peering Engineering (EPE)

- ISIS-SR with MPLS in the Fabric (reachability to loopbacks)
- Peering Router advertises BGP-LU label for connected EBGP peer interface to controller
- ADD-PATH/BMP provides the BGP controller with visibility to all BGP paths

Controller

- BGP-LU to steer traffic for specific prefixes to egress interface with nexthop corresponding to the Peering router.
- Resolve over the ISIS_SR MPLS transport tunnel

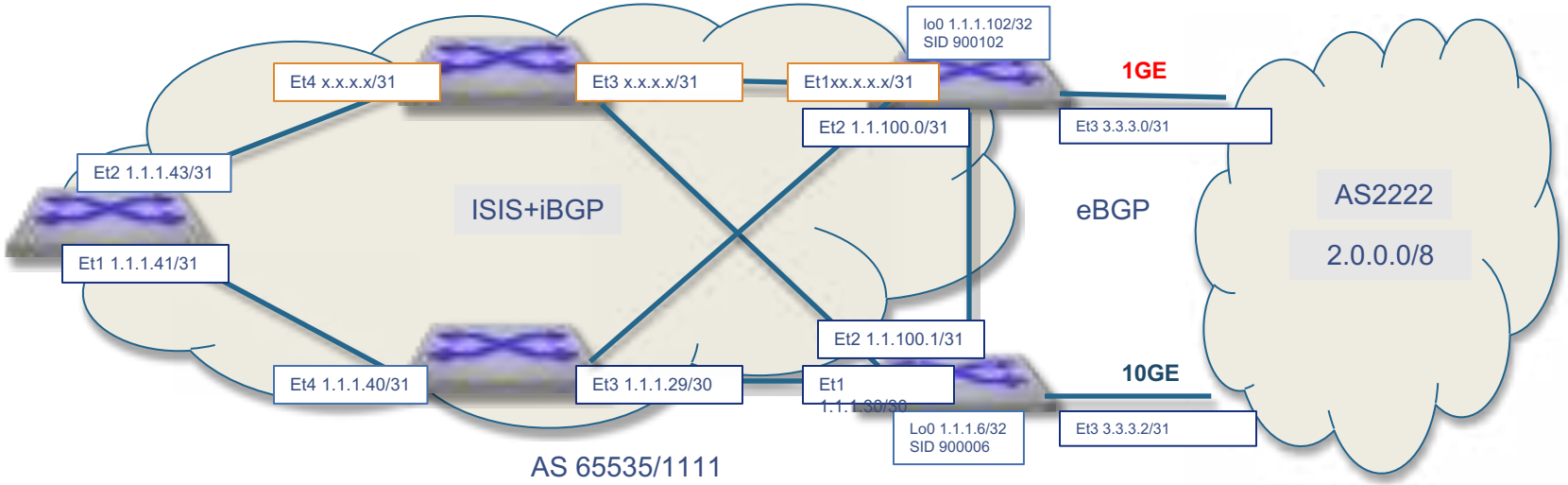
Bandwidth & Path

UCMP+Segment-Routing

ARISTA

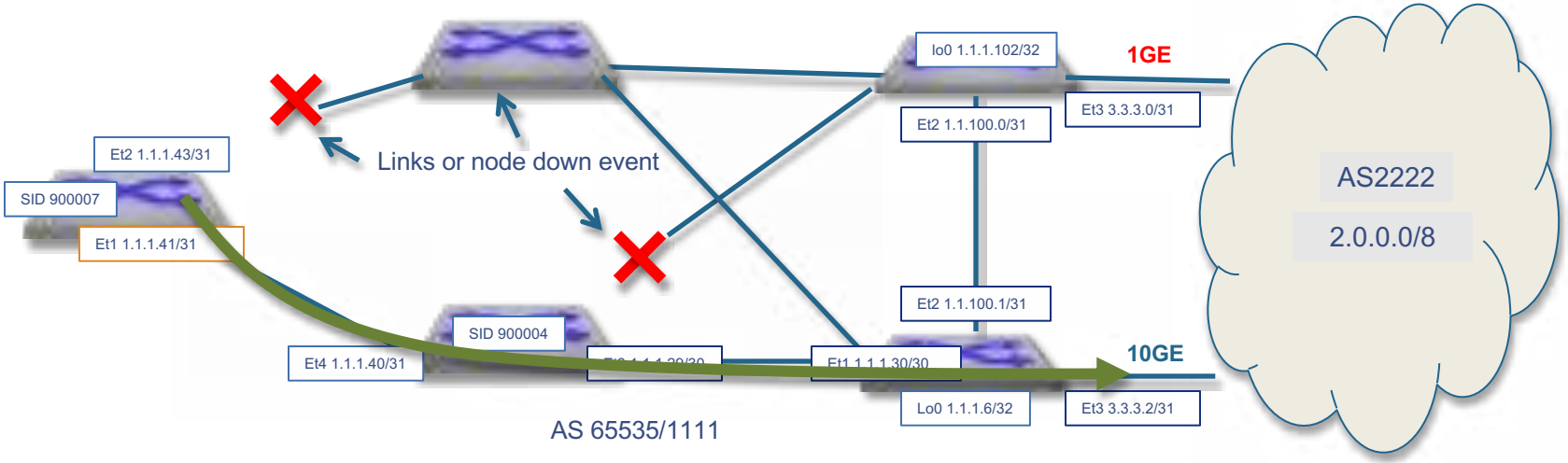
BGP UCMP&Segment-routing, the perfect storm

- ECMP peering with mixed peering speeds and propagate LBW EXTENDED_COMMUNITIES down to the Leafs
- The path control however is with Segment-Routing



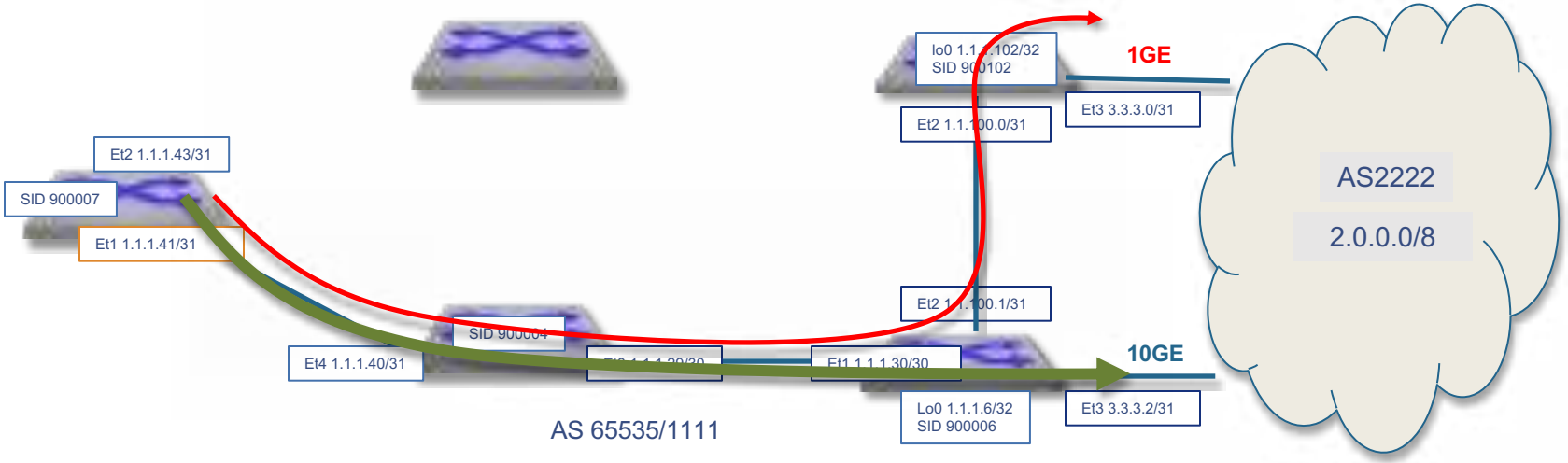
Scenario #1

- Peering are bandwidth and price critical.
- Even if topology (due to link/node down) no longer provide ECMP, expensive peers needs to be used. Bw inside the network not problematic
- However if IP only forwarding, all traffic will egress on only one path



BGP UCMP & Segment-routing

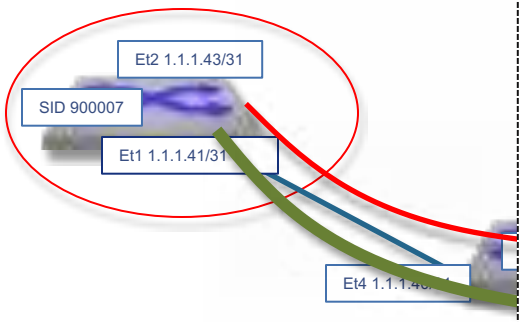
- Hashing towards both peers handle by UCMP
- Segment-routing secure traffic goes to both BGP Edges and not only to the 10GE Peer, since forwarding is based on labels (900102&900006)



BGP UCMP in a Segment-routing network…

Recursive Lookup

- Pls Follow the BGP NEXT_HOP double recursion

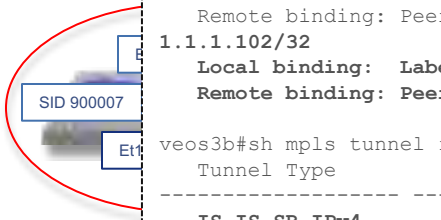



```
Veos3b#sh ip bgp 2.0.0.0
BGP routing table information for VRF default
Router identifier 1.1.1.7, local AS number 64515
BGP routing table entry for 2.0.0.0/8
 Paths: 2 available
  2222
    3.3.3.3 from 1.1.1.6 (1.1.1.6)
      Origin INCOMPLETE, metric 0, localpref 100, IGP metric 0, weight 0, received 00:05:24 ago, valid,
      internal, ECMP head, ECMP, UCMP, best, ECMP contributor
      Extended Community: Link-Bandwidth-AS:65535:10000000000.000000
      Rx SAFI: Unicast
  2222
    3.3.3.0 from 1.1.1.102 (1.1.1.102)
      Origin INCOMPLETE, metric 0, localpref 100, IGP metric 0, weight 0, received 00:05:24 ago, valid,
      internal, ECMP, UCMP, ECMP contributor
      Extended Community: Link-Bandwidth-AS:65535:1000000000.000000
      Rx SAFI: Unicast

veos3b#sh ip bgp 3.3.3.3
BGP routing table entry for 3.3.3.2/31
 Paths: 1 available
  Local
    1.1.1.6 from 1.1.1.6 (1.1.1.6)
      Origin IGP, metric 0, localpref 100, IGP metric 30, weight 0, received 00:06:38 ago, valid,
      internal, best
      Rx SAFI: Unicast

veos3b#sh ip bgp 3.3.3.0
BGP routing table entry for 3.3.3.0/31
 Paths: 1 available
  Local
    1.1.1.102 from 1.1.1.102 (1.1.1.102)
      Origin IGP, metric 0, localpref 100, IGP metric 40, weight 0, received 00:06:50 ago, valid,
      internal, best
      Rx SAFI: Unicast

veos3b#sh ip ro 1.1.1.6
 I L2    1.1.1.6/32 [115/30] via 1.1.1.40, Ethernet1

veos3b#sh ip ro 1.1.1.102
 I L2    1.1.1.102/32 [115/40] via 1.1.1.40, Ethernet1
```

BGP UCMP in a Segment-routing network…

MPLS Lookup

- Ingress node MPLS table

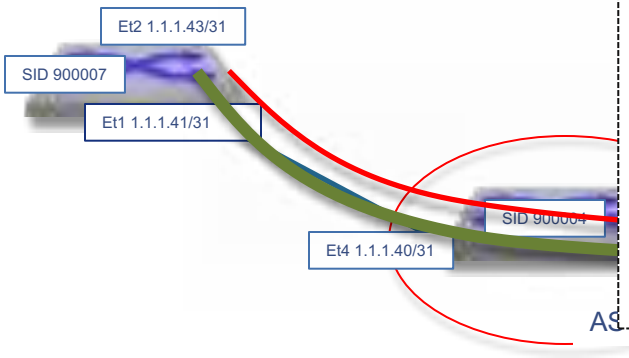



```
veos3b#sh mpls segment-routing bi
1.1.1.4/32
   Local binding:  Label: 900004
   Remote binding: Peer ID: 0010.0100.1004, Label: imp-null
1.1.1.6/32
   Local binding:  Label: 900006
   Remote binding: Peer ID: 0010.0100.1004, Label: 900006
1.1.1.7/32
   Local binding:  Label: imp-null
   Remote binding: Peer ID: 0010.0100.1004, Label: 900007
1.1.1.102/32
   Local binding:  Label: 900102
   Remote binding: Peer ID: 0010.0100.1004, Label: 900102

veos3b#sh mpls tunnel fib
   Tunnel Type          Index       Endpoint           Nexthop        Interface      Labels
------------------- ----------- ------------------ -------------- ----------------- ----------
   IS-IS SR IPv4        3           1.1.1.102/32       1.1.1.40       'Ethernet1'    [ 900102 ]
   IS-IS SR IPv4        1           1.1.1.4/32         1.1.1.40       'Ethernet1'    [ 3 ]
   IS-IS SR IPv4        2           1.1.1.6/32         1.1.1.40       'Ethernet1'    [ 900006 ]
(…)
```

BGP UCMP in a Segment-routing network…

Transit node MPLS table

- Since traffic towards BGP_Edge with NEXT_HOP 3.3.3.0 and SID 900102 will SWAP on the Transit node, it will be label forwarded whole path and thereby traffic will not just enter the first BGP_edge box with SID 900006
- Result in both peerings being used

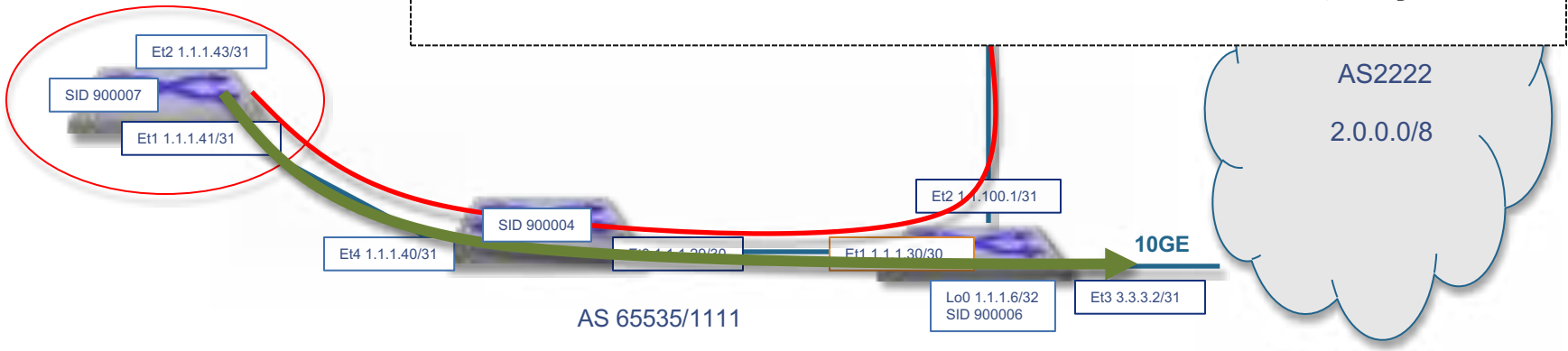



```
veos4#sh mpls lfib route
MPLS forwarding table (Label [metric] Vias) - 5 routes
MPLS next-hop resolution allow default route: False
Via Type Codes:
          M - Mpls Via, P - Pseudowire Via,
          I - IP Lookup Via, V - Vlan Via,
          VA - EVPN Vlan Aware Via, ES - EVPN Ethernet Segment Via,
          VF - EVPN Vlan Flood Via, AF - EVPN Vlan Aware Flood Via
Source Codes:
          S - Static MPLS Route, B2 - BGP L2 EVPN,
          B3 - BGP L3 VPN, P - Pseudowire,
          L - LDP, IP - IS-IS SR Prefix Segment,
          IA - IS-IS SR Adjacency Segment, IL - IS-IS SR Segment to LDP,
          LI - LDP to IS-IS SR Segment, BL - BGP LU,
          DE - Debug LFIB
(…)
IP  900006    [1], 1.1.1.6/32
                via M, 1.1.1.30, pop
                    payload autoDecide, ttlMode autoDecide, apply egress-acl
                    interface Ethernet3
 IP  900102    [1], 1.1.1.102/32
                via M, 1.1.1.30, swap 900102
                    payload autoDecide, ttlMode autoDecide, apply egress-acl
                    interface Ethernet3
(…)
```

BGP UCMP in a Segment-routing network…

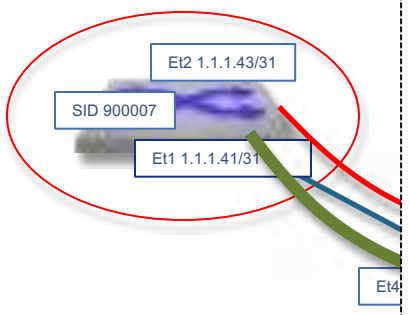
- BGP Weights applied

```
veos3b#sh ip ro 2.0.0.0

VRF: default
Codes: C - connected, S - static, K - kernel,
       O - OSPF, IA - OSPF inter area, E1 - OSPF external type 1,
       E2 - OSPF external type 2, N1 - OSPF NSSA external type 1,
       N2 - OSPF NSSA external type2, B I - iBGP, B E - eBGP,
       R - RIP, I L1 - IS-IS level 1, I L2 - IS-IS level 2,
       O3 - OSPFv3, A B - BGP Aggregate, A O - OSPF Summary,
       NG - Nexthop Group Static Route, V - VXLAN Control Service,
       DH - Dhcp client installed default route

 B I     2.0.0.0/8 [100/0] via 1.1.1.40, Ethernet1 label 900006, weight 10/11
                           via 1.1.1.40, Ethernet1 label 900102, weight 1/11
```



BGP UCMP in a Segment-routing network…

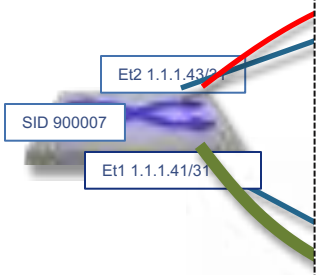
- The very same principle can be applied to generated routes, for example a default route which is commonly in use from BGP-Edge down in the Datacenter
- Of course the design can be full-feed down to servers, and just labels for NEXT_HOP in th




```
veos3b#  sh ip bgp 0.0.0.0
BGP routing table information for VRF default
Router identifier 1.1.1.7, local AS number 64515
BGP routing table entry for 0.0.0.0/0
 Paths: 2 available
  Local
    3.3.3.3 from 1.1.1.6 (1.1.1.6)
      Origin INCOMPLETE, metric 0, localpref 100, IGP metric 0, weight 0, received 00:02:16 ago,
      valid, internal, ECMP head, ECMP, UCMP, best, ECMP contributor
      Extended Community: Link-Bandwidth-AS:65535:10000000000.000000
      Rx SAFI: Unicast
  Local
    3.3.3.0 from 1.1.1.102 (1.1.1.102)
      Origin INCOMPLETE, metric 0, localpref 100, IGP metric 0, weight 0, received 00:02:16 ago,
      valid, internal, ECMP, UCMP, ECMP contributor
      Extended Community: Link-Bandwidth-AS:65535:1000000000.000000
      Rx SAFI: Unicast

veos3b#sh ip ro 0.0.0.0/0
(...)
Gateway of last resort:
 B I    0.0.0.0/0 [100/0] via 1.1.1.40, Ethernet1 label 900006, weight 10/11
                          via 1.1.1.40, Ethernet1 label 900102, weight 1/11
```

Scenario #2

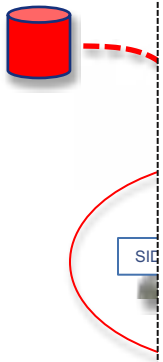
- For more specific path engineering, BGP-LU can be used as programming tool from the controller, then its careless of topology or metric
- Consider default ECMP scenario towards Transit Peering




```
veos3b(config)#sh ip bgp 0.0.0.0
BGP routing table information for VRF default
Router identifier 1.1.1.7, local AS number 64515
BGP routing table entry for 0.0.0.0/0
 Paths: 2 available
  Local
    3.3.3.3 from 1.1.1.6 (1.1.1.6)
      Origin IGP, metric 0, localpref 100, IGP metric 0, weight 0, received 01:09:55 ago, valid, internal,
      ECMP head, ECMP, UCMP, best, ECMP contributor
      Extended Community: Link-Bandwidth-AS:65535:10000000000.000000
      Rx SAFI: Unicast
  Local
    3.3.3.0 from 1.1.1.102 (1.1.1.102)
      Origin IGP, metric 0, localpref 100, IGP metric 0, weight 0, received 01:09:53 ago, valid, internal,
      ECMP, UCMP, ECMP contributor
      Extended Community: Link-Bandwidth-AS:65535:1000000000.000000
      Rx SAFI: Unicast

veos3b(config)#sh ip ro 0.0.0.0
(…)
Gateway of last resort:
 B I    0.0.0.0/0 [200/0] via 1.1.1.40, Ethernet1 label 900006, weight 10/11
                          via 1.1.1.42, Ethernet2 label 900102, weight 1/11
```

BGP UCMP in a Segment-routing network…

- S
- T
  p
- A
  c




```
lunkan@ubunt:~$ /usr/sbin/exabgp /etc/exabgp/exabgp.conf
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | generate it using "exabgp --fi > /usr/etc/exabgp/exabgp.env"
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | reactor       | Performing reload of exabgp 3.4.12
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading | #
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading | # IPv4&Label-unicast template
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading | neighbor 192.168.10.101 {
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |    router-id 192.168.10.70;
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |    group-updates true;
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |    local-address 192.168.10.70;
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |    local-as 64515;
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |    peer-as 64515;
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |    hold-time 180;
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |    capability {
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |       graceful-restart 120;
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |       route-refresh;
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |       add-path send;
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |    }
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |    family {
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |       ipv4 nlri-mpls;
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |    }
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |    static {
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |       route 2.0.0.0/8 {
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |          path-information 0.0.0.1;
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |          next-hop 1.1.1.40;
 Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |           label [ 900004 900006 ];
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |        }
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |       route 2.0.0.0/8 {
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |          path-information 0.0.0.2;
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |          next-hop 1.1.1.42;
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |          label [ 900005 900102 900006];
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |        }
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading |    }
Mon, 19 Feb 2018 18:50:51 | INFO     | 1735   | configuration | loading | }
```

BGP UCMP in a Segment-routing network

- The
  prog
  NEX
  weig
  rout

```
veos3b(config)#sh ip bgp labeled-unicast det
BGP routing table information for VRF default
Router identifier 1.1.1.7, local AS number 64515
BGP routing table entry for 2.0.0.0/8
 Paths: 2 available
  Local
    1.1.1.40 labels [ 900004 900006 ] from 192.168.10.70 (192.168.10.70)
      Origin IGP, metric 0, localpref 100, IGP metric 1, weight 0, received 01:35:38 ago, valid, internal,
      ECMP head, best, ECMP contributor
      Rx path id: 0x1
      Rx SAFI: Labels
  Local
    1.1.1.42 labels [ 900005 900102 900006 ] from 192.168.10.70 (192.168.10.70)
      Origin IGP, metric 0, localpref 100, IGP metric 1, weight 0, received 01:35:38 ago, valid, internal,
      ECMP, ECMP contributor
      Rx path id: 0x2
      Rx SAFI: Labels

veos3b(config)#sh ip ro 2.0.0.0/8
(…)
 B I     2.0.0.0/8 [200/0] via 1.1.1.40, Ethernet1 label 900004 900006
                           via 1.1.1.42, Ethernet2 label 900005 900102 900006
```