FORTINET®

Secure SD-WAN design

One-Size-Fits-All

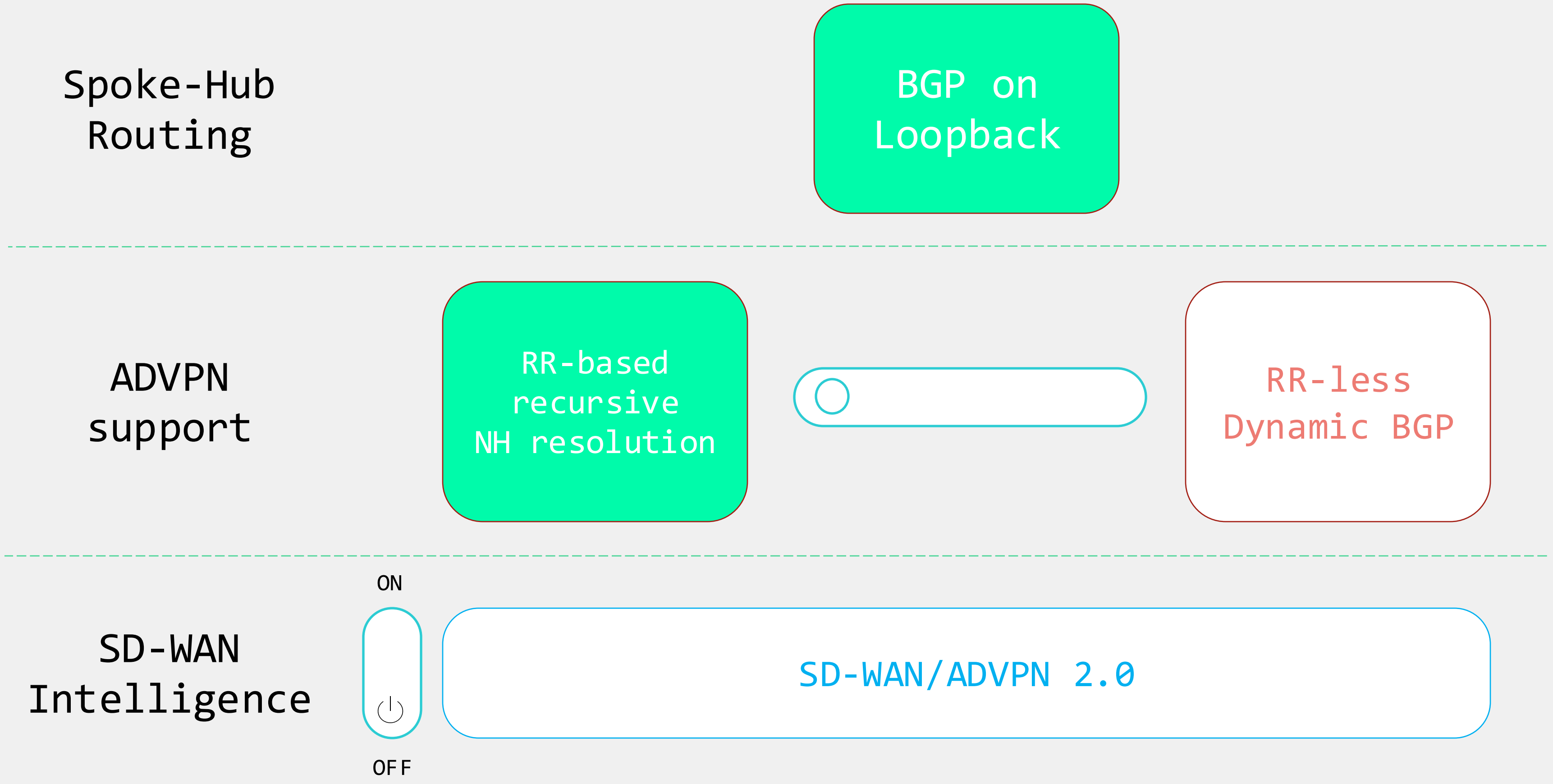
The Art of One-Size-Fits-All

Where do we stand and where are we going?

Overlay Network

Design Principles:

- Layered approach
- Each layer has its own duties
- Features can be switched on/off without affecting other layers



Spoke-Hub Routing

BGP on Loopback

ADVPN support

RR-based recursive NH resolution

RR-less Dynamic BGP

SD-WAN Intelligence

ON

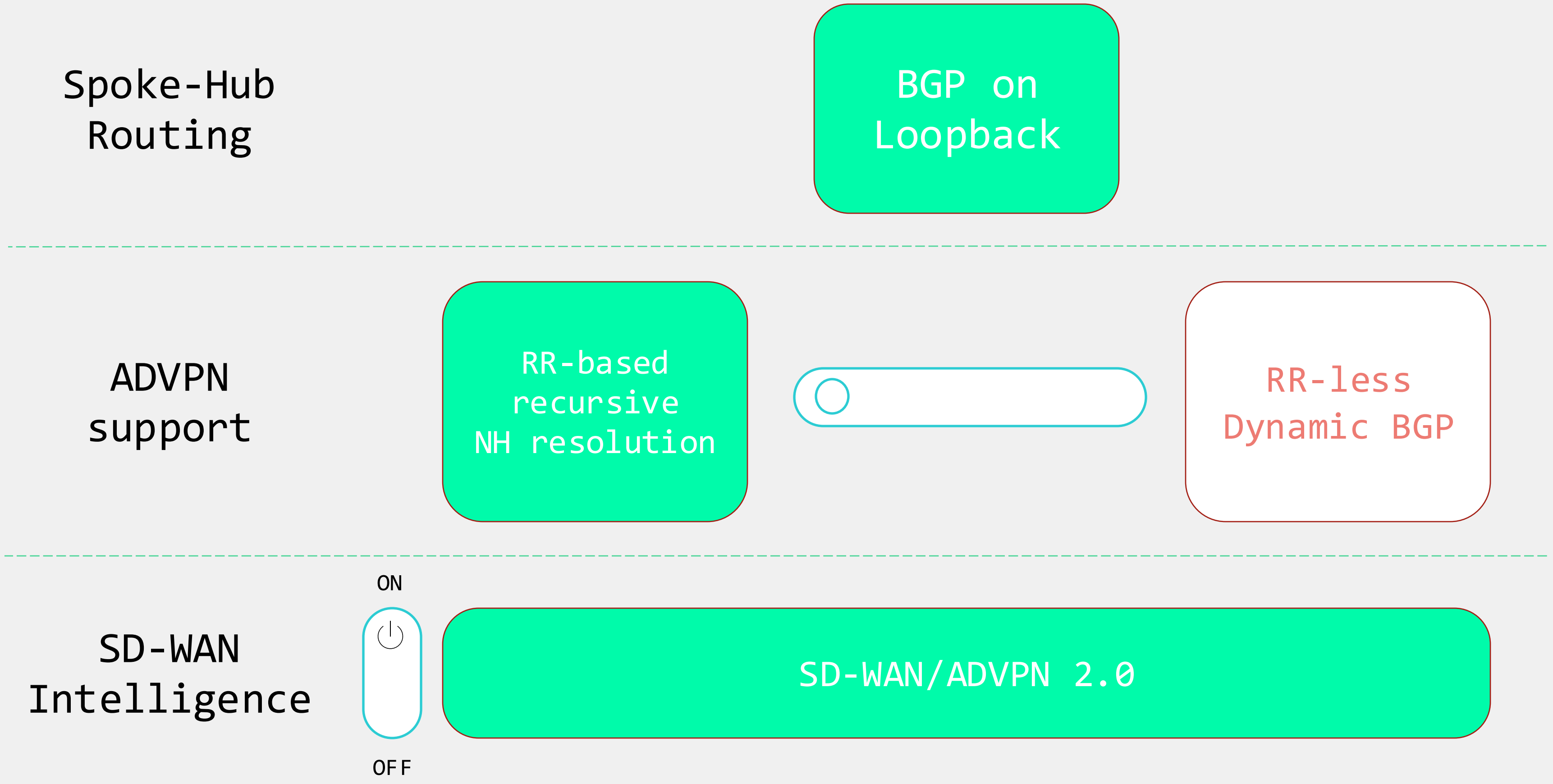
OFF

SD-WAN/ADVPN 2.0

Overlay Network

Design Principles:

- Layered approach
- Each layer has its own duties
- Features can be switched on/off without affecting other layers



Spoke-Hub Routing

BGP on Loopback

ADVPN support

RR-based recursive NH resolution

RR-less Dynamic BGP

SD-WAN Intelligence

ON

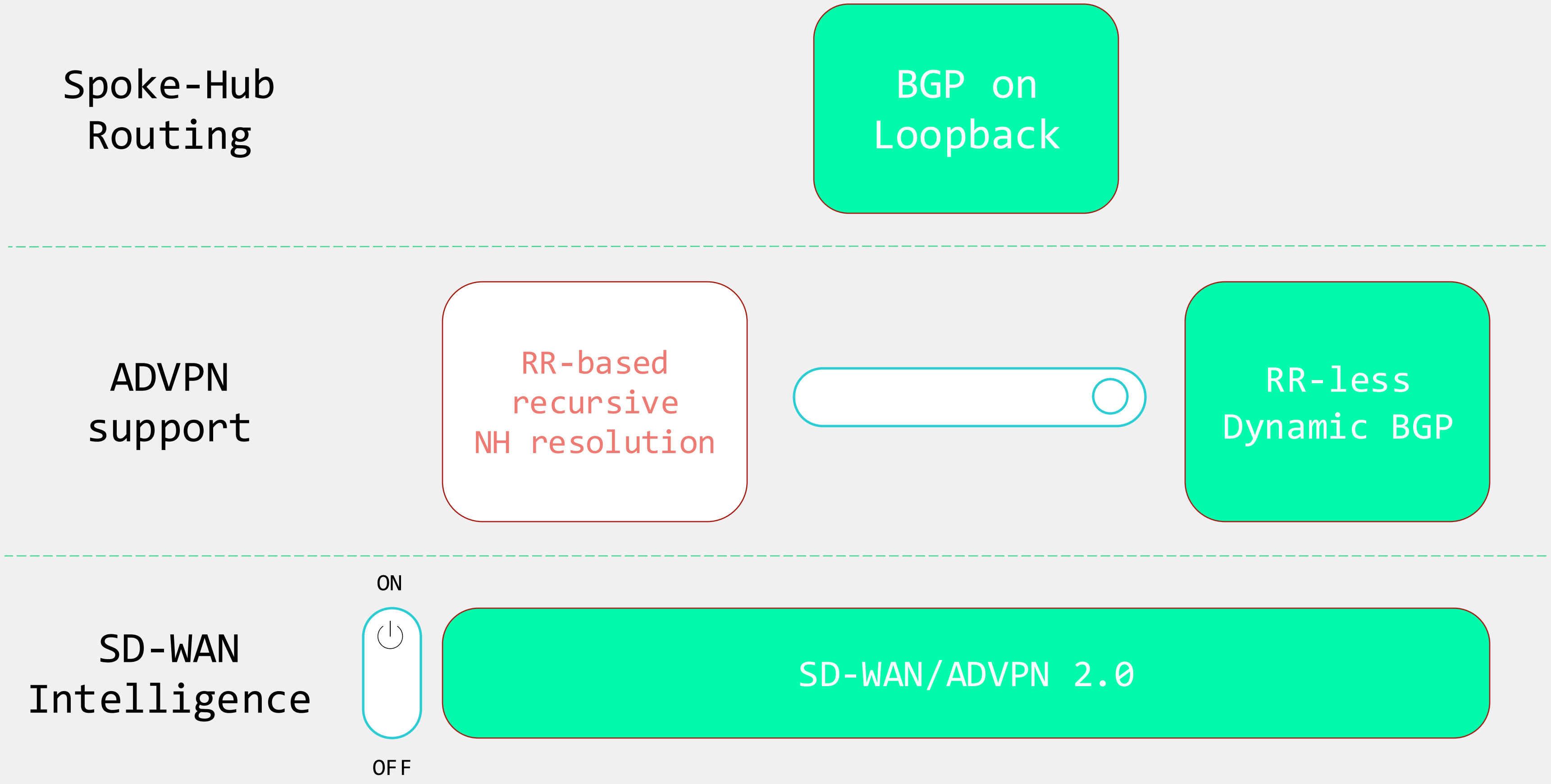
OFF

SD-WAN/ADVPN 2.0

Overlay Network

Design Principles:

- Layered approach
- Each layer has its own duties
- Features can be switched on/off without affecting other layers



Spoke-Hub Routing

BGP on Loopback

ADVPN support

RR-based recursive NH resolution

RR-less Dynamic BGP

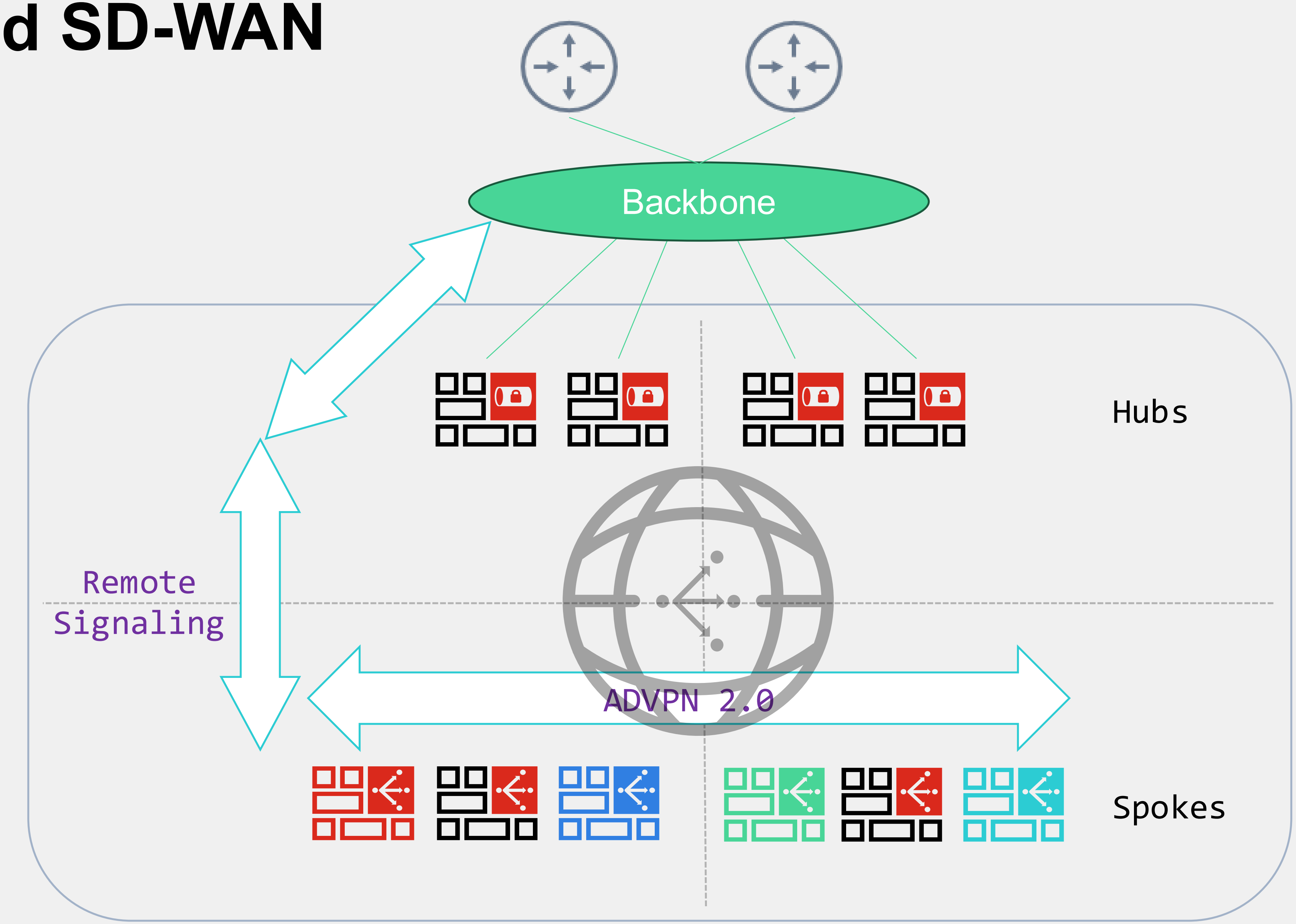
SD-WAN Intelligence

ON

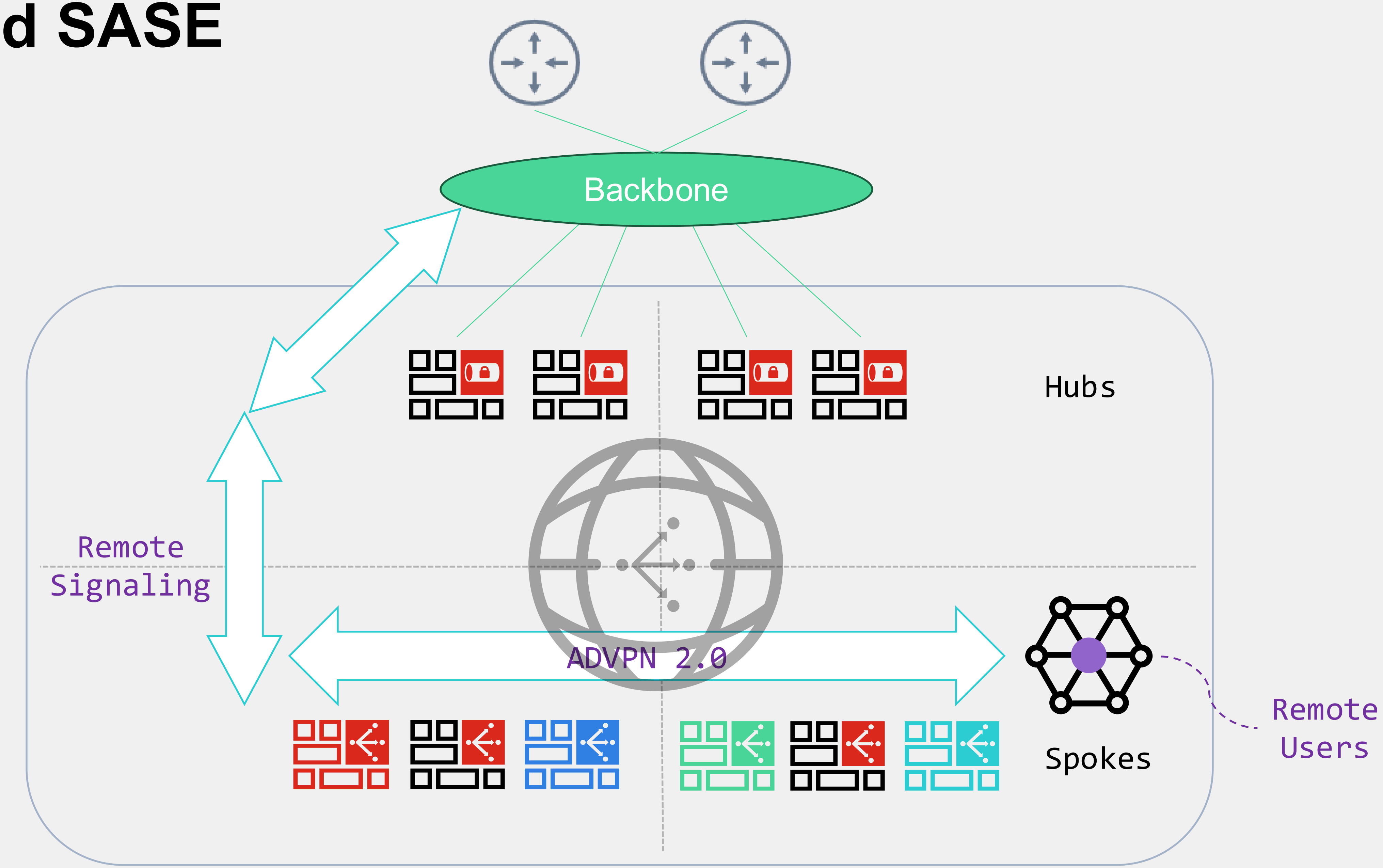
OFF

SD-WAN/ADVPN 2.0

Unified SD-WAN



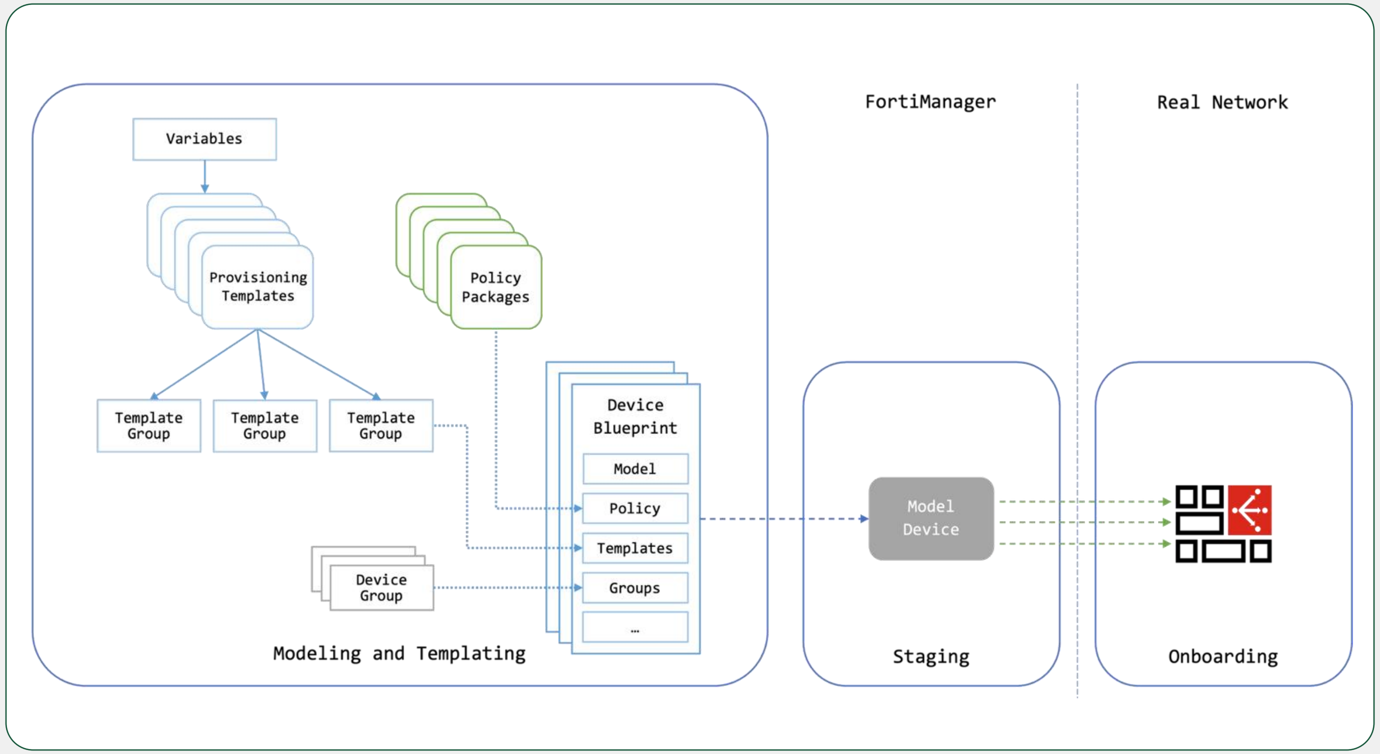
Unified SASE



Deployment Workflow

Our familiar principles remain unchanged:

- Strive to have 100% templated configuration
  - FortiManager = Source of Truth
- Make Templates as generic as possible, using variables
  - Reusable on different types of sites
  - Reusable in different projects
- Automate!



Deployment Workflow

Our familiar principles remain unchanged:

- Strive to have 100% templated configuration
  - FortiManager = Source of Truth
- Make Templates as generic as possible, using variables
  - Reusable on different types of sites
  - Reusable in different projects
- Automate!

- Onboard a new tenant as follows:
  - Clone your “Master ADOM” with all the Templates
  - OR create the new ADOM using Automation
  - Onboard tenant sites
- Onboard a new site as follows:
  - Create a Model Device, assigning it to the right Device Group(s) and setting its variables
  - Install policy and configuration on the Model Device
  - Link the real device using your preferred method (ZTP, LTP…)

FortiManager Toolset

Also in Release 7.4, our recommended approach for **Telco/MSSP** remains unchanged:

- The new release of the **Jinja Orchestrator** supports the complete “golden package”
- Additionally, it unifies single-VRF and multi-VRF flavors, allows mixed RR-based/RR-less deployments with “BGP on Loopback” and (much) more.

	FMG Tools
Underlay (Interfaces, IPs…)	Jinja Orchestrator
Overlay (IPSEC)	
Routing (BGP)	
SD-WAN (ADVPN 2.0)	SD-WAN Templates
Firewall Policies	Policy Packages

SD-WAN/ADVPN 2.0

The Native SD-WAN Intelligence

In a Nutshell

The **SD-WAN/ADVPN 2.0** framework is a new generation of ADVPN designed for SD-WAN and natively integrated with it.

Its main control-plane mechanisms are:

1. **Discovery.** The originating node discovers the remote node. It learns about its topology and the current health status of all its participating SD-WAN Members.
2. **Path Selection**. After the discovery, the originating node combines the local and the remote data, selects an optimal shortcut and triggers it.
3. **Health Updates.** Periodic health updates are sent over the active shortcuts, allowing the Path Selection to revisit its previous choice on per-rule basis, possibly triggering new shortcuts.

It is designed to be almost “plug-and-play” – just enable it and let it work.

Configuration

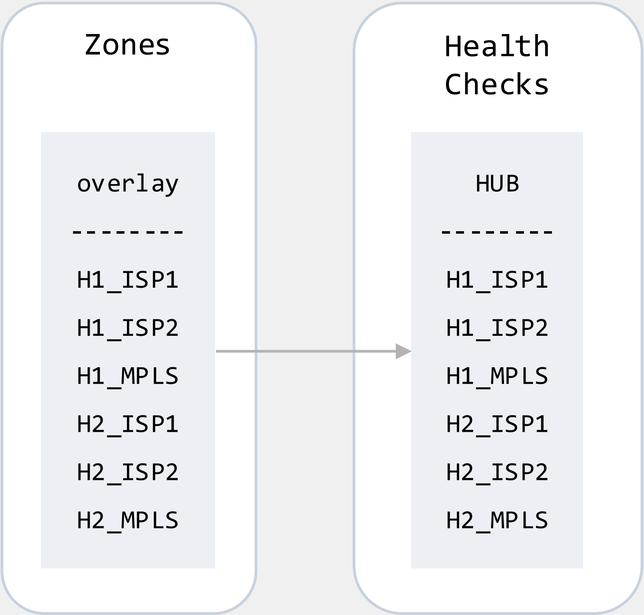
```
config system sdwan
  config zone
    edit "overlay"
      set advpn-select enable
      set advpn-health-check "HUB"
    next
  end
end
```

```
config system sdwan
  config members
    edit “H1_ISP1”
      set transport-group 1
    next
    edit ”H1_MPLS"
      set transport-group 2
    next
  end
end
```

- ADVPN 2.0 is enabled under the SD-WAN Zone.
- The scope of Discovery is determined by the assigned Health Check

- Segregated transports are identified by different **transport-groups** assigned to SD-WAN Members.

Configuration



Zones

overlay

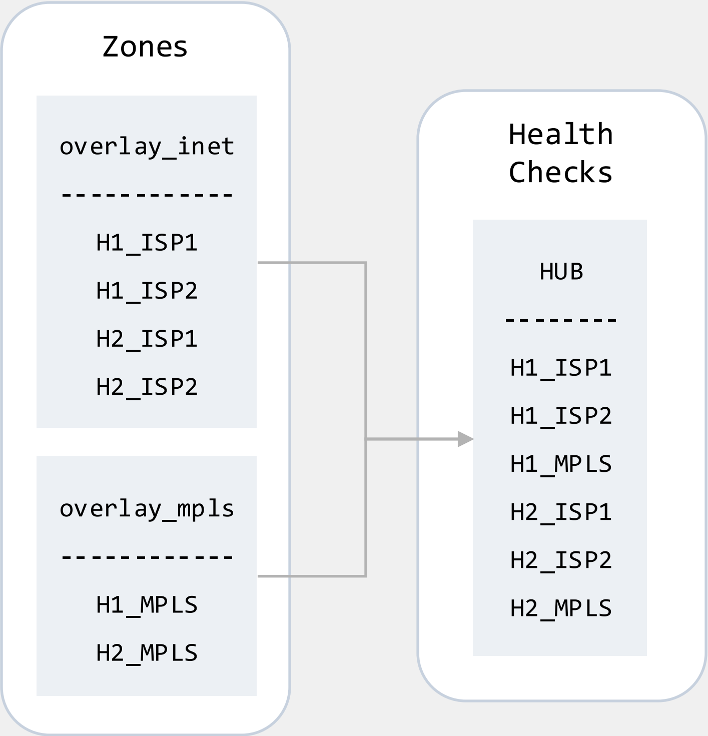
H1_ISP1
H1_ISP2
H1_MPLS
H2_ISP1
H2_ISP2
H2_MPLS

Health Checks

HUB

H1_ISP1
H1_ISP2
H1_MPLS
H2_ISP1
H2_ISP2
H2_MPLS

Single Zone + Single HC
for all overlays and Hubs



Zones

overlay_inet

H1_ISP1
H1_ISP2
H2_ISP1
H2_ISP2

overlay_mpls

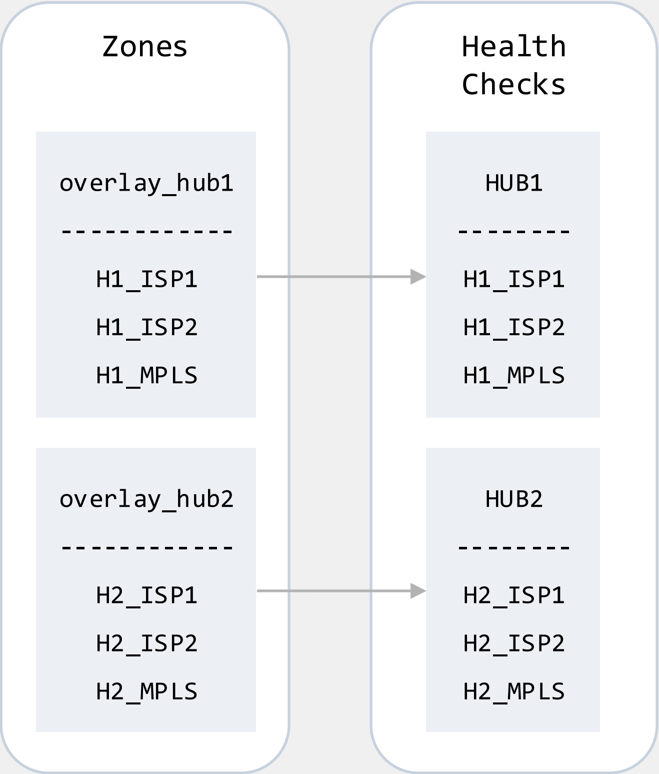
H1_MPLS
H2_MPLS

Health Checks

HUB

H1_ISP1
H1_ISP2
H1_MPLS
H2_ISP1
H2_ISP2
H2_MPLS

Multiple Zones, Single HC



Zones

overlay_hub1

H1_ISP1
H1_ISP2
H1_MPLS

overlay_hub2

H2_ISP1
H2_ISP2
H2_MPLS

Health Checks

HUB1

H1_ISP1
H1_ISP2
H1_MPLS

HUB2

H2_ISP1
H2_ISP2
H2_MPLS

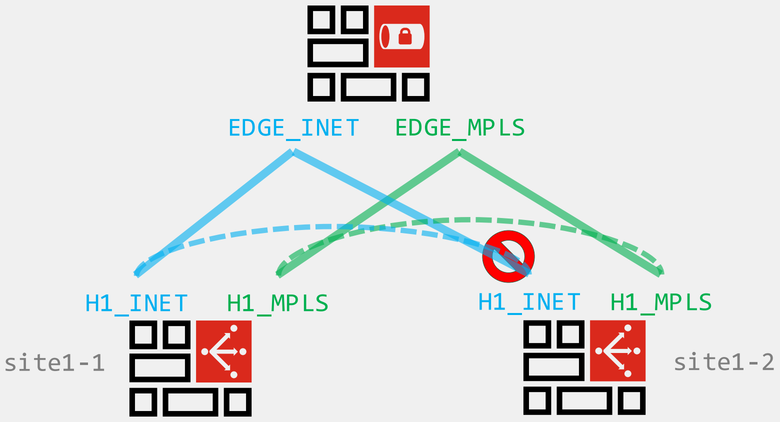
Separate HC per Hub

ADVPN 2.0 Use Cases

The Intelligence in Action

Segregated Transports (e.g. INET+MPLS)

1. The user traffic goes to the Hub via H1_INET (`s11->s12`)
   - This triggers Discovery
   - Now `s11` learns that `s12` does not have a member in `tg.1`
2. Path Selection on `s11` decides to trigger H1_MPLS_0 shortcut
3. Later, INET link recovers on `s12`
4. Recovery kicks in:
   - Triggered by a Health Update
   - New healthy member in `tg.1` is detected on `s12`
   - Path Selection on `s11` decides to trigger H1_INET_0 shortcut



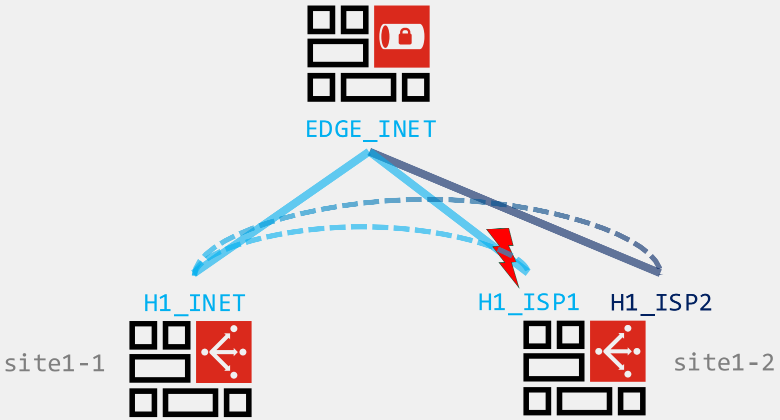
```
config members
  edit “H1_INET”
    set transport-group 1
  next
  edit ”H1_MPLS"
    set transport-group 2
  next
end
```

```
config service
  edit ”Corporate”
    set mode sla
    set priority-members H1_INET H1_MPLS
  next
end
```

```
Corporate   H1_INET_0 H1_MPLS_0 H1_INET H1_MPLS
```

Choosing Remote Internet Link

1. The user traffic goes to the Hub via H1_INET (s11->s12)
    - This triggers Discovery
    - Both members on s12 belong to the same tg.
2. Path Selection on s11 decides to trigger H1_INET_0 shortcut
    - Towards ISP1 link on s12
3. Later, ISP1 link becomes unhealthy on s12
4. Recovery kicks in:
    - Triggered by a Health Update
    - Path Selection on s11 decides to trigger H1_INET_1 shortcut, from the same local link (INET) to a different remote link (ISP2)



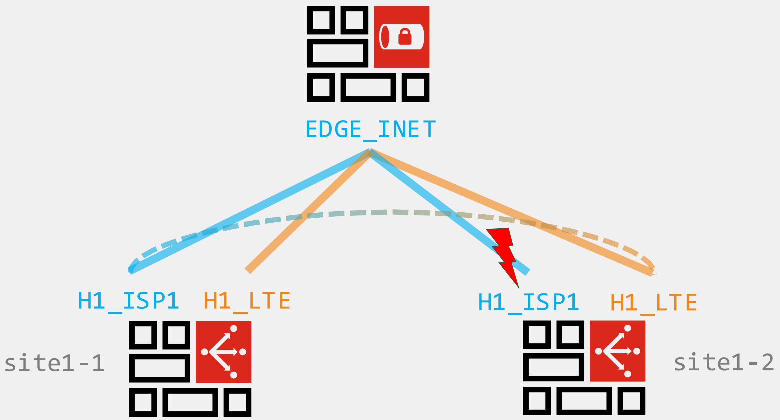
```
config service
  edit ”Corporate”
    set mode sla
    set priority-members H1_INET
  next
end
```

```
Corporate   H1_INET_0 H1_INET
```

Backup of Last Resort

In this example, ISP1 is a broadband link, while LTE is a backup of last resort (billed by traffic volume; thus, to be avoided)

1. The user traffic goes to the Hub via H1_ISP1 (s11->s12)
   - This triggers Discovery
   - Both members on s12 have the same tg., but different **costs**
   - Now s11 learns that ISP1 link is unhealthy on s12
2. Path Selection on s11 decides to trigger H1_ISP1_0 shortcut
   - Towards the **LTE** link on s12
   - The LTE link on s11 is **not used**



```
config vpn ipsec phase1-interface
  edit ”H1_LTE"
    set link-cost 10
  next
end
```

```
config members
  edit ”H1_LTE"
    set cost 10
  next
end
```

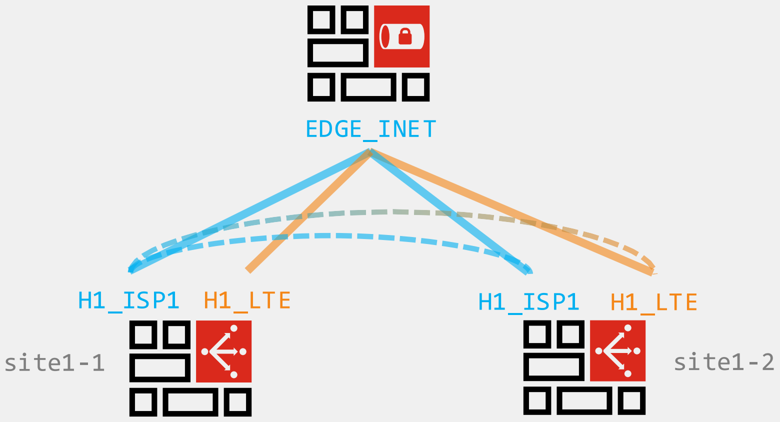
```
config service
  edit ”Corporate”
    set mode sla
    set priority-members H1_ISP1 H1_LTE
  next
end
```

Corporate	H1_ISP1_0 H1_ISP1 H1_LTE

Backup of Last Resort

In this example, ISP1 is a broadband link, while LTE is a backup of last resort (billed by traffic volume; thus, to be avoided)

3. Later, ISP1 link recovers on `s12`
4. Recovery kicks in:
   - Triggered by a Health Update
   - Path Selection on `s11` decides to trigger H1_ISP1_1 shortcut, towards **ISP1** link on `s12`, to stop using LTE
   - The new shortcut H1_ISP1_1 is inserted **before** the old one, thanks to the lower **remote** `link-cost` (sent by `s12`)



```
config vpn ipsec phase1-interface
  edit ”H1_LTE"
    set link-cost 10
  next
end
```

```
config members
  edit ”H1_LTE"
    set cost 10
  next
end
```

```
config service
  edit ”Corporate”
    set mode sla
    set priority-members H1_ISP1 H1_LTE
  next
end
```

```
Corporate    H1_ISP1_1  H1_ISP1_0  H1_ISP1  H1_LTE
```

Per-Service Recovery

1. App-A traffic starts
   - Path Selection creates a H1_INET_0 shortcut
   - App-B rule is idle, no action needed
2. App-B traffic starts
   - Traffic matches the existing H1_INET_0 shortcut in App-B rule
3. **Per-service recovery** (a.k.a. per-rule recovery) kicks in:
   - Triggered by a Health Update
   - New hit is detected in the App-B rule
   - Path Selection decides to trigger H1_MPLS_0 shortcut

```
config system sdwan
  config service
    edit ”App-A”
      set mode sla
      set priority-members H1_INET H1_MPLS
    next
    edit “App-B”
      set mode sla
      set priority-members H1_MPLS H1_INET
    next
  end
end
```

App-A	H1_INET_0 H1_INET H1_MPLS
App-B	H1_INET_0 H1_MPLS H1_INET

Per-Service Recovery

1. App-A traffic starts
   - Path Selection creates a H1_INET_0 shortcut
   - App-B rule is idle, no action needed
2. App-B traffic starts
   - Traffic matches the existing H1_INET_0 shortcut in App-B rule
3. **Per-service recovery** (a.k.a. per-rule recovery) kicks in:
   - Triggered by a Health Update
   - New hit is detected in the App-B rule
   - Path Selection decides to trigger H1_MPLS_0 shortcut

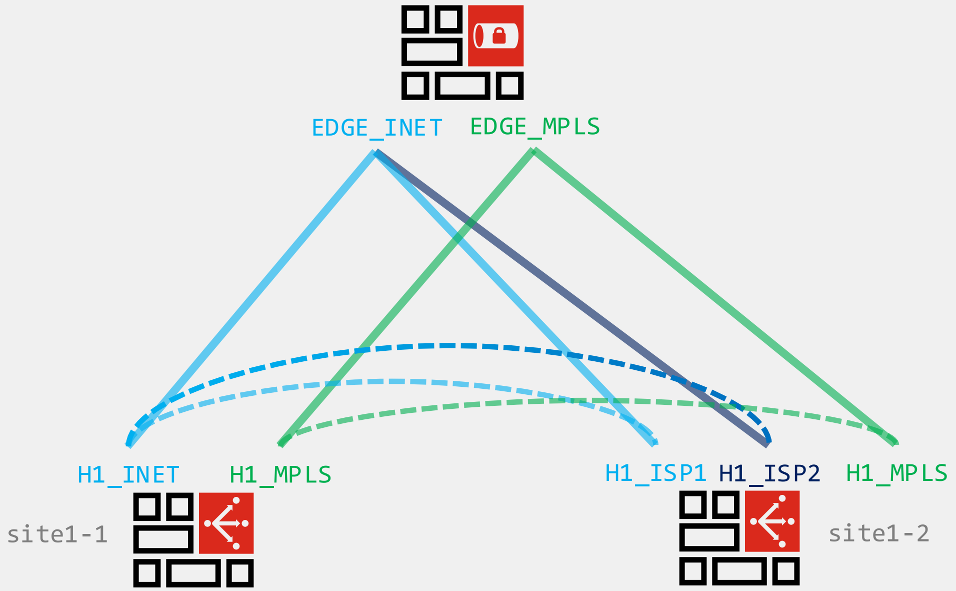
```
config system sdwan
  config service
    edit ”App-A”
      set mode sla
      set priority-members H1_INET H1_MPLS
    next
    edit “App-B”
      set mode sla
      set priority-members H1_MPLS H1_INET
    next
  end
end
```

App-A	H1_INET_0 H1_MPLS_0 H1_INET H1_MPLS
App-B	H1_MPLS_0 H1_INET_0 H1_MPLS H1_INET

Load-Balancing (7.6.1+)

Trigger all shortcuts at once between the two Spokes and load-balance between them, respecting the SLA.

- ADVPN Shortcut Monitoring guarantees accurate end-to-end health measurement



```
config system sdwan
  config service
    edit ”Corporate”
      set mode sla
      set load-balance enable
      set priority-members H1_INET H1_MPLS
    next
  end
end
```

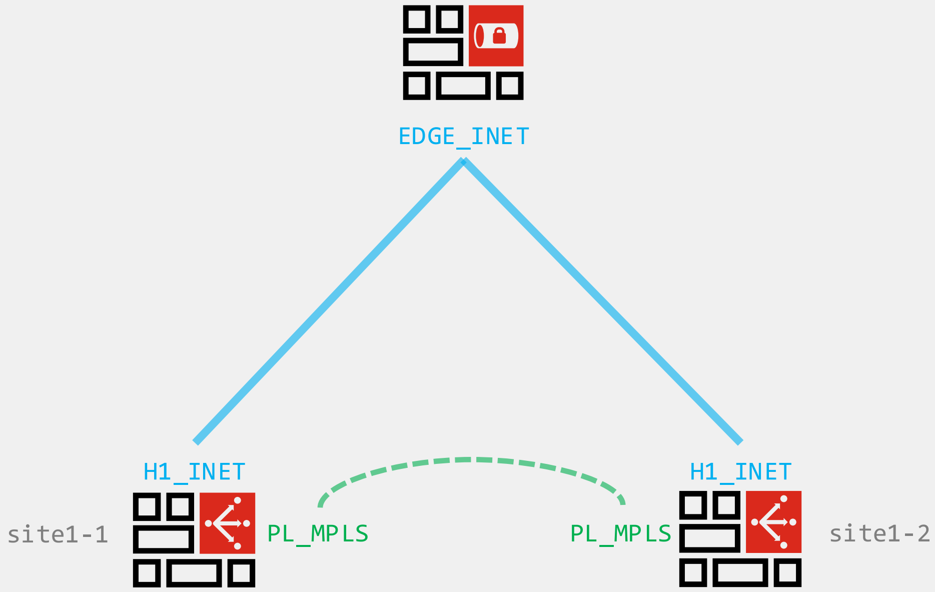
Overlay Placeholders (7.6.1+)

Build Spoke-to-Spoke shortcuts over transports to which the Hubs are not connected:

- Decouple Spoke-to-Spoke connectivity from Spoke-to-Hub
- Examples: Hubs deployed in Public Cloud / MSSP Cloud

How does it work?

- Configure the "placeholders" as ADVPN 2.0 members
- Exchange their transport-groups during Discovery
- Trigger all possible shortcuts using placeholders
  - NOTE: No Spoke-to-Hub health estimation is possible!
- ADVPN Shortcut Monitoring guarantees accurate end-to-end health measurement (once the shortcuts are up)



ADVPN 2.0 Mechanisms

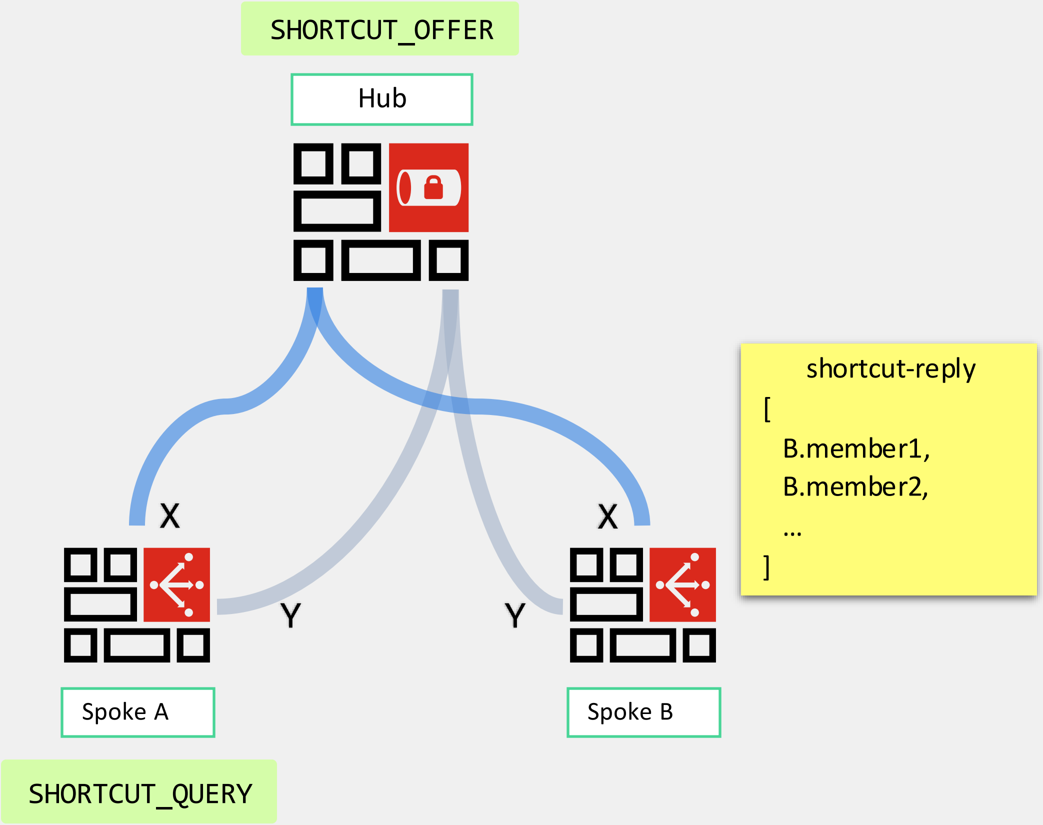
Deep-Dive for Better Understanding

Discovery

1. The user traffic goes to the Hub (`sA->sB`)
2. The Hub sends SHORTCUT_OFFER to `sA`
3. The local Spoke (`sA`) sends SHORTCUT_QUERY
   - It must reach the remote Spoke *somehow*
4. The remote Spoke (`sB`) replies with SHORTCUT_REPLY
   - Including full **Discovery** information

✓ Overlay Stickiness is *abolished*.

B.memberX = (wan_ip, cost, health, transport_group…)



shortcut-reply
[
B.member1,
B.member2,
…
]

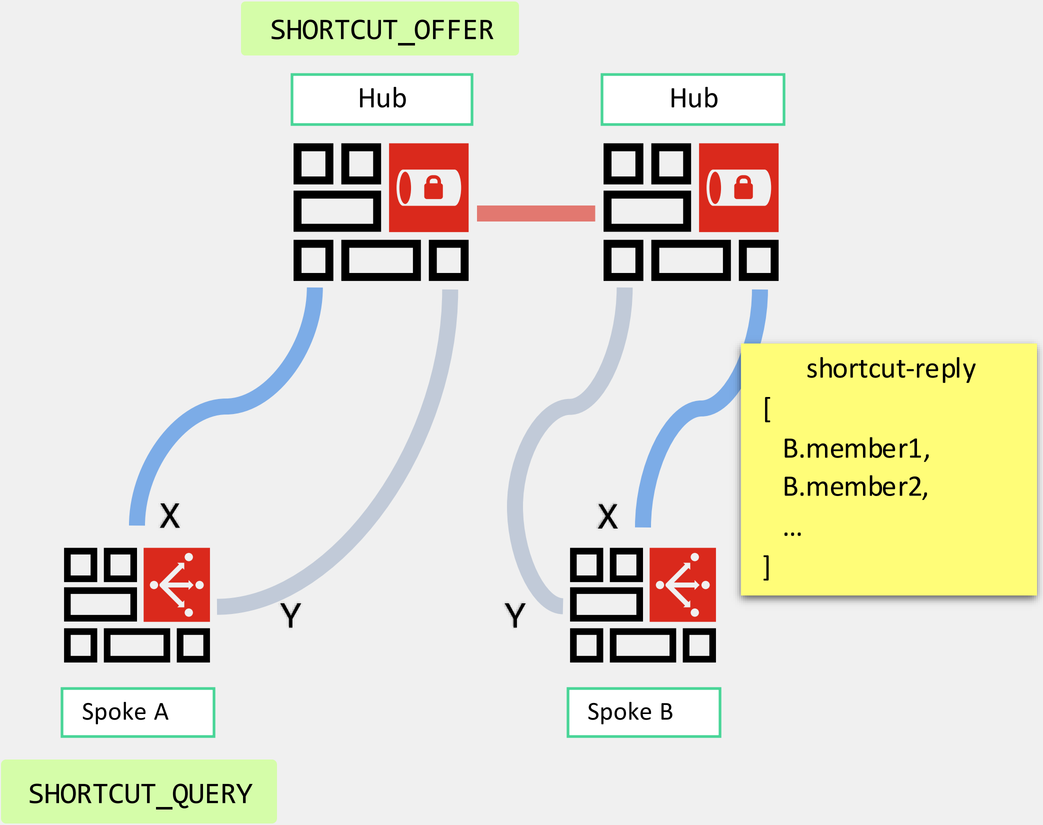
Discovery

1. The user traffic goes to the Hub (`sA->sB`)
2. The Hub sends SHORTCUT_OFFER to `sA`
3. The local Spoke (`sA`) sends SHORTCUT_QUERY
   - It must reach the remote Spoke *somehow*
4. The remote Spoke (`sB`) replies with SHORTCUT_REPLY
   - Including full **Discovery** information

✓ Overlay Stickiness is *abolished*.

✓ Native multi-regional support.

B.memberX = (wan_ip, cost, health, transport_group…)

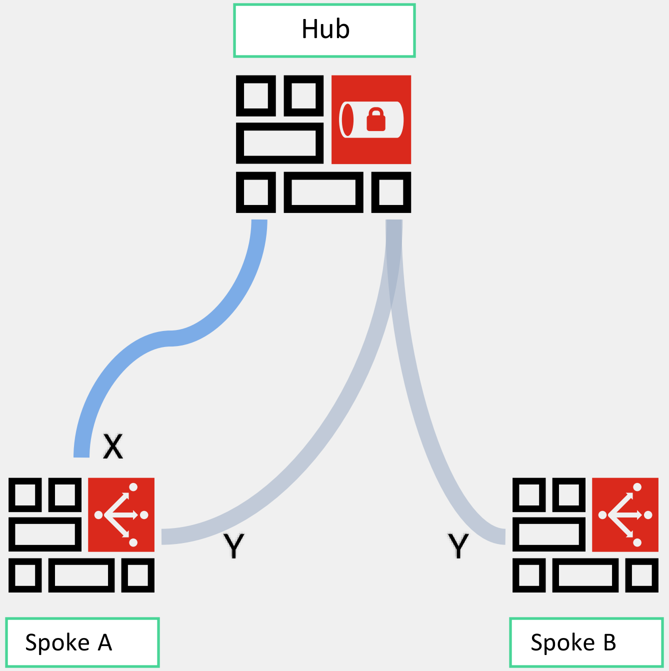


shortcut-reply
[
B.member1,
B.member2,
…
]

Discovery

Example:

- The Overlay X is not available on Spoke B (either permanently or temporarily)
- After the Discovery, Spoke A is aware of that!



```
shortcut-reply
[
 B.member2 = {
   sla_ok = 0x1,
   transport_group = 2
  }
]
```

Path Selection

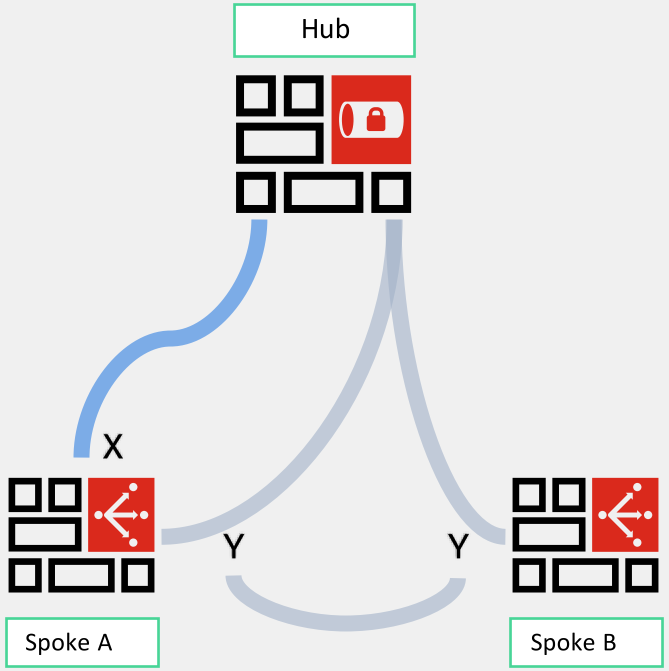
The originating node *locally* selects both ends of the shortcut and triggers IKE negotiation accordingly. The path selection depends on:

- Local members info
- Remote members info
- Mode of the matched SD-WAN rule

```
Local info:
[
 A.member1 = {
  sla_ok = 0x1,
  transport_group = 1
 },
 A.member2 = {
  sla_ok = 0x1,
  transport_group = 2
 }
]
```

+

```
Remote info:
[
 B.member2 = {
  sla_ok = 0x1,
  transport_group = 2
 }
]
```



Path Selection

For “mode sla”:

1. Select the most preferred “left” end which is in-SLA
2. Try to match a “right” end in the same transport-group which is also in-SLA
   - If you can, then path selection is complete
   - If you cannot, try the next preferred “left” end

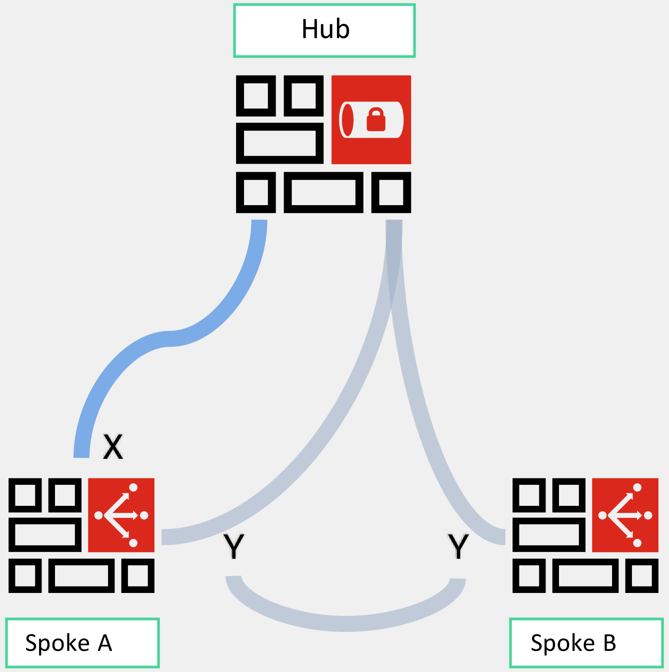
For “mode priority”:

1. Select “left” and “right” ends in the same transport-group with the best possible combined health

```
Local info:
[
  A.member1 = {
    sla_ok = 0x1,
    transport_group = 1
  },
  A.member2 = {
    sla_ok = 0x1,
    transport_group = 2
  }
]
```

+

```
Remote info:
[
  B.member2 = {
    sla_ok = 0x1,
    transport_group = 2
  }
]
```



Health Updates

Periodic health updates will be sent over the active shortcuts

- Every 5 seconds

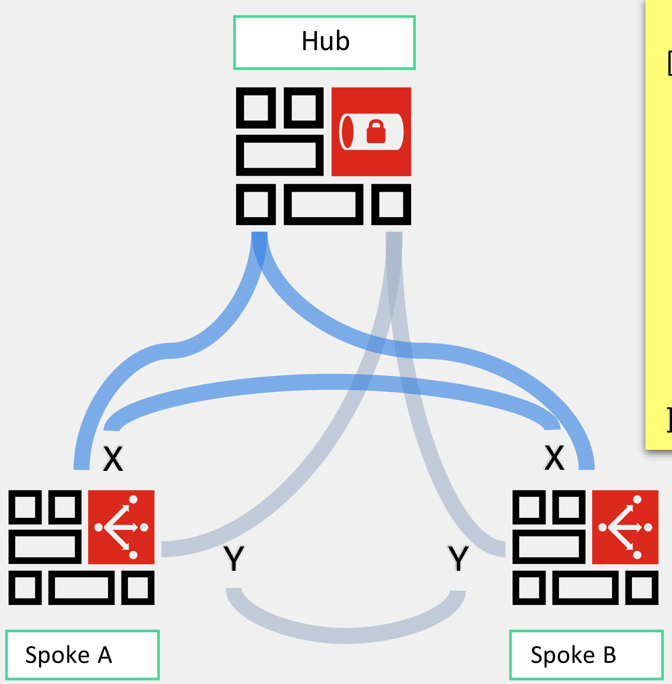
Upon receiving an update, the node will *revisit* the path selection on per-rule basis

- Only for the rules currently hit by traffic

If necessary, new shortcuts will be triggered.

Local info:

```
[
 A.member1 = {
  sla_ok = 0x1,
  transport_group = 1
 },
 A.member2 = {
  sla_ok = 0x1,
  transport_group = 2
 }
]
```



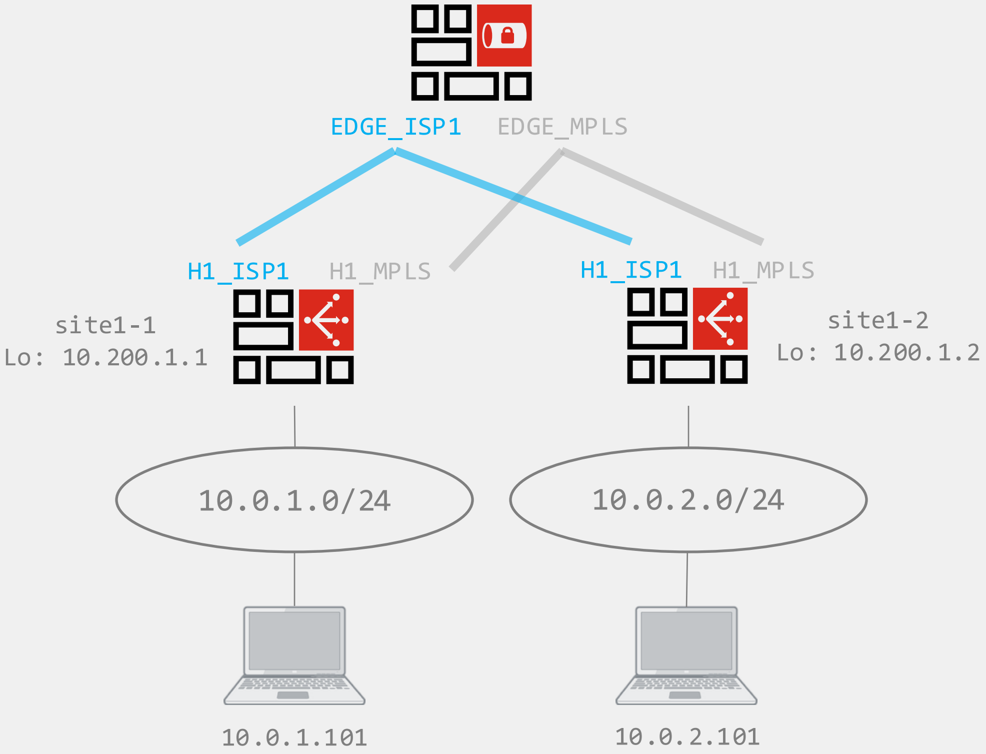
Remote info:

```
[
 B.member1 = {
  sla_ok = 0x1,
  transport_group = 1
 },
 B.member2 = {
  sla_ok = 0x1,
  transport_group = 2
 }
]
```

Shortcut Creation

Hub-assisted Shortcut Creation

1. The user traffic goes to the Hub (s11->s12)
2. The Hub sends SHORTCUT_OFFER to s11
3. The local Spoke (s11) sends SHORTCUT_QUERY
   - Including **user traffic** src/dst IP
4. The remote Spoke (s12) replies with SHORTCUT_REPLY
   - Including full **Discovery** information
5. Path Selection decides which shortcut to create and updates the remote Spoke with SHORTCUT_UPDATE
   - Facilitates NAT support



```
H1_ISP1: shortcut-offer 10.0.1.101->10.0.2.101 …
```

```
H1_ISP1: send shortcut-query 10895447131714993536 …
10.0.1.101->10.0.2.101
```

```
H1_ISP1: recv shortcut-reply 10895447131714993536 …
to 10.0.1.101
```

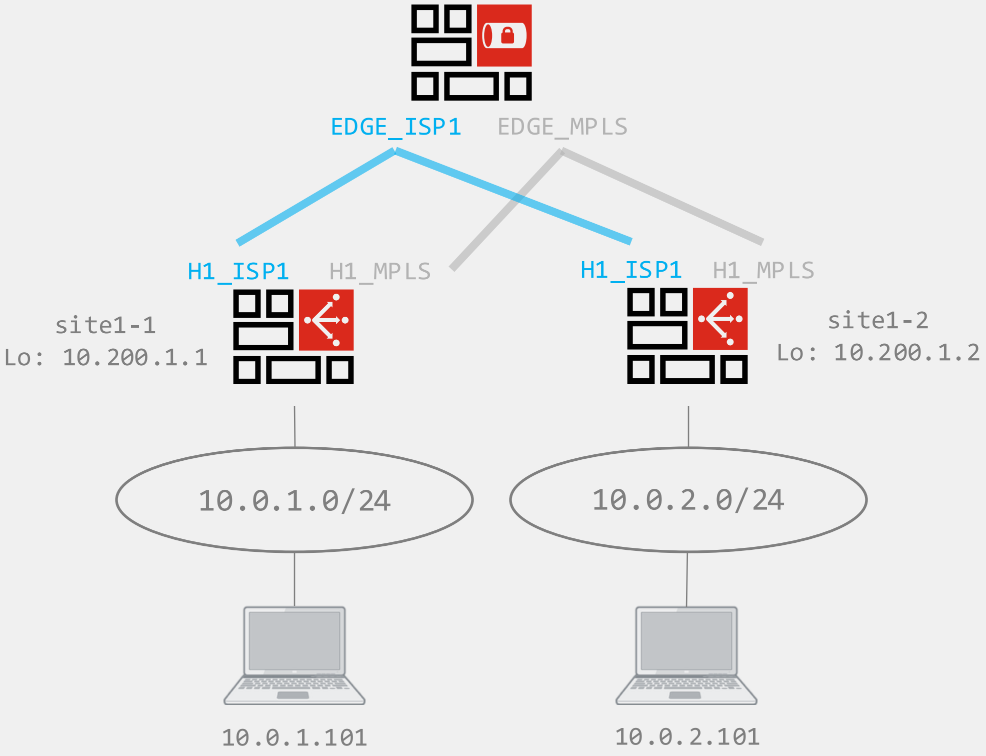
```
send vwl oif request (0xa2fb9f39) for intf H1_ISP1 site site1-2-0-overlay
recv vwl advpn oif response (0xa2fb9f39)
vwl oif result: port4

H1_ISP1: queue shortcut-update 10895447131714993536 … daddr 10.0.2.101
ext-addr 172.16.0.1 … resp-gw H1_MPLS
```

Shortcut Creation

Direct Shortcut Creation

1. Path Selection decides to trigger a new shortcut
   - Due to changes in health conditions
2. The local Spoke (s11) sends SHORTCUT_QUERY
   - Including **loopback** src/ip IP
3. The remote Spoke (s12) replies with SHORTCUT_REPLY
4. The local Spoke (s11) updates the remote Spoke with SHORTCUT_UPDATE
   - Facilitates NAT support



```
:VWL_ADVPN_MSG_T_TRIGGER
H1_ISP1: send shortcut-query 16524642307507723332 …
10.200.1.1->10.200.1.2
```

```
H1_ISP1: recv shortcut-reply 16524642307507723332 …
to 10.200.1.1
```

```
H1_ISP1: queue shortcut-update 16524642307507723332 … daddr 10.200.1.2
ext-addr 192.2.0.1 … resp-gw H1_ISP1
```

Shortcut Creation

So which mechanism will kick in…?

- For the very first shortcut between the two Spokes – always **Hub-assisted shortcut creation**.
- For subsequent shortcuts – it depends…
  - Scenario #1:
    - A link goes out of SLA on remote Spoke
    - Within 5 sec. the local Spoke receives a Health Update about it
    - Path Selection decides to trigger another shortcut
    - Result: **direct shortcut creation**
  - Scenario #2:
    - A link goes out of SLA on remote Spoke
    - Shortcut Monitoring detects that and moves the shortcut to the end of the proute oif list
    - User traffic uses a valid route to the Hub via one of the overlays
    - The Hub sends SHORTCUT_OFFER
    - Path Selection decides what shortcut to trigger this time
    - Result: **Hub-assisted shortcut creation**

Multi-hop Paths (non-shortcuts)

At this stage, ADVPN 2.0 is totally focused on **shortcuts**.

- It is assumed that whenever a shortcut can be built, you want to use it
- The **shortcut priority** mode is enabled by default with ADVPN 2.0 and it is required for its correct operation.
  - *Prefer (in-sla) shortcuts over all (in-sla) parents*

User traffic can still flow via the Hubs (multi-hop), but this is not controlled by the Path Selection.

- The existing proute matching mechanism remains unchanged: when there is no in-sla shortcut, a parent tunnel can be selected, assuming that there is a valid route to the destination (or best, if “tie-break fib-best-match” is used)
- This is where your routing design can make a difference!
  - Do you need tag-match? Do you need a default route via the overlays?

Routing Design

The ADVPN 2.0 is designed to be independent from the routing design.

None of the described control-plane mechanisms depend on any BGP tweak, none of them prohibit the use of any protocol feature (such as route summarization). We are free to choose the routing design which will perform best with regards to its direct duties – the route advertisement.

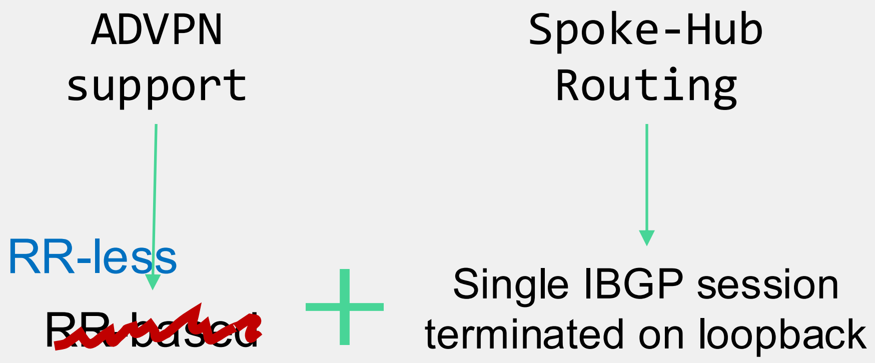
What are these duties?

- Reachability for all the **user traffic** across the entire SD-WAN overlay network. In the other words, the LAN prefixes must be somehow advertised.
- Reachability for all the **loopbacks** across the entire SD-WAN overlay network. This is one of the few requirements that the ADVPN 2.0 does put on the routing design.
  - For example, it is required for the correct operation of the **direct shortcut creation** mechanism.

Dynamic BGP on Loopback

RR-less ADVPN Design

In a Nutshell



ADVPN support

Spoke-Hub Routing

RR-less

~~RR-based~~ + Single IBGP session terminated on loopback

Dynamic BGP on Loopback

Dynamic BGP on Loopback

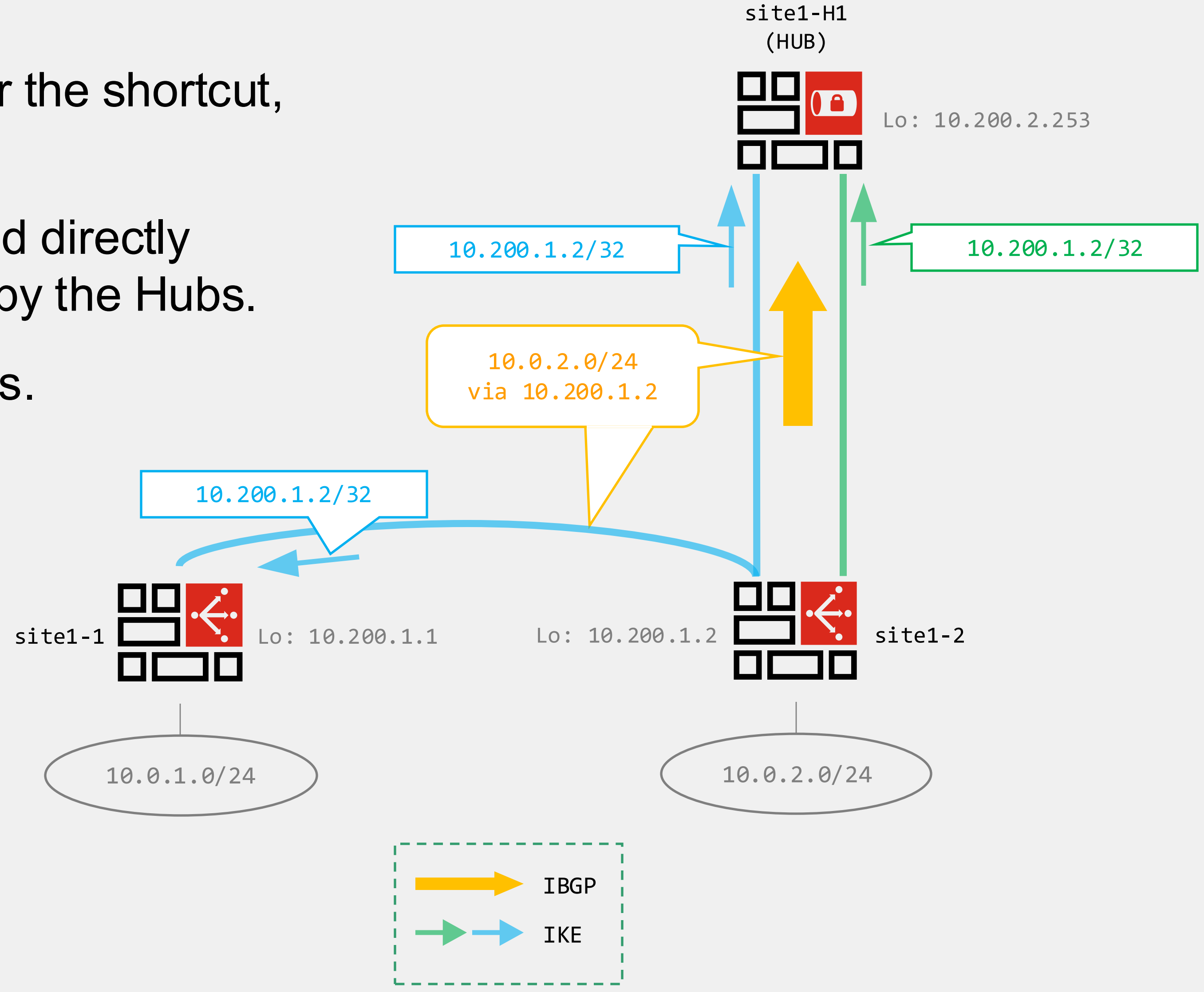
IKE dynamically triggers a BGP session over the shortcut, between the exchanged IPs (loopbacks).

RR-less design: LAN prefixes are advertised directly over the shortcut, instead of being reflected by the Hubs.

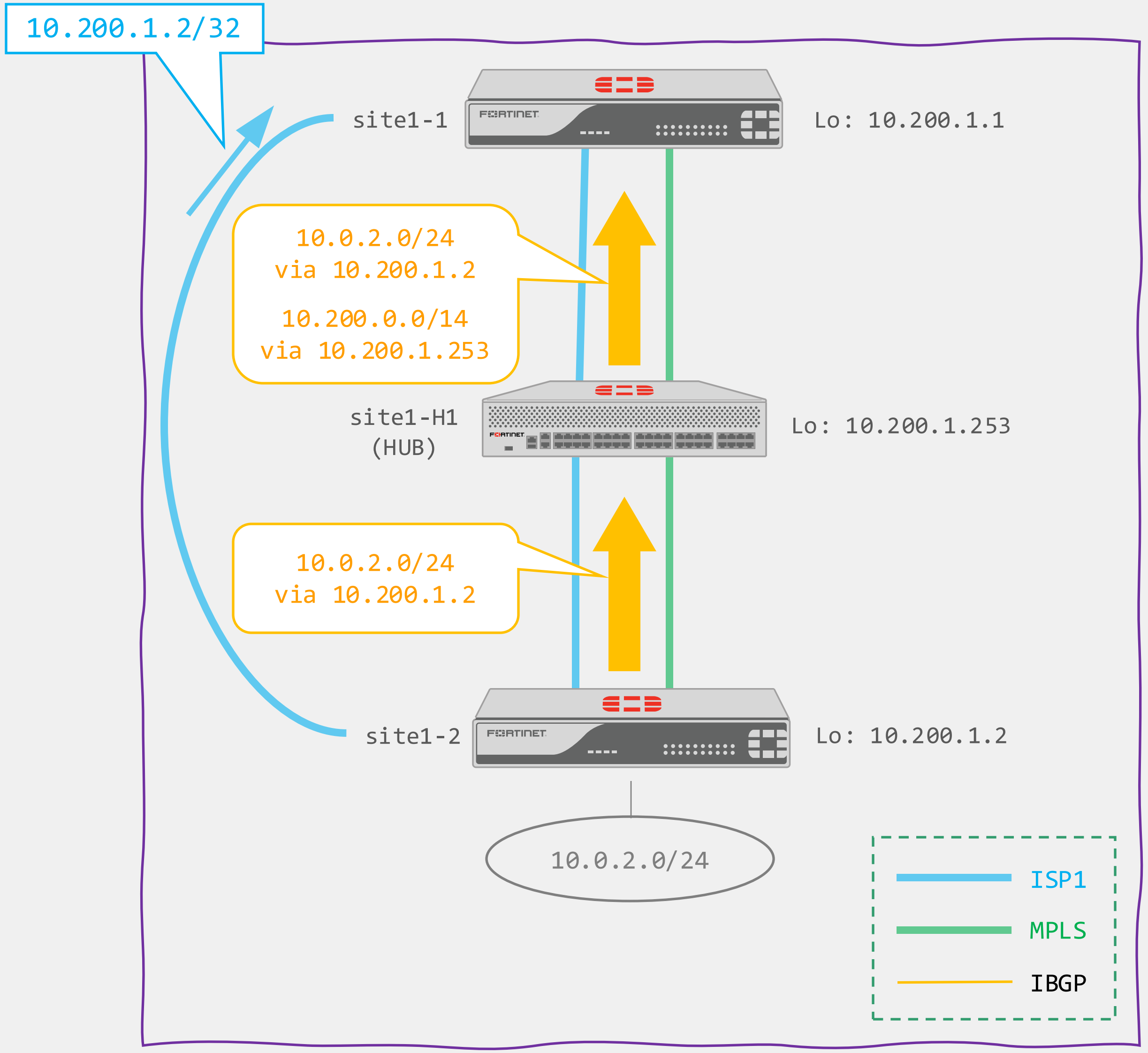
Route summarization is **allowed** on the Hubs.

Supports IPv6, VPNv4, VPNv6…

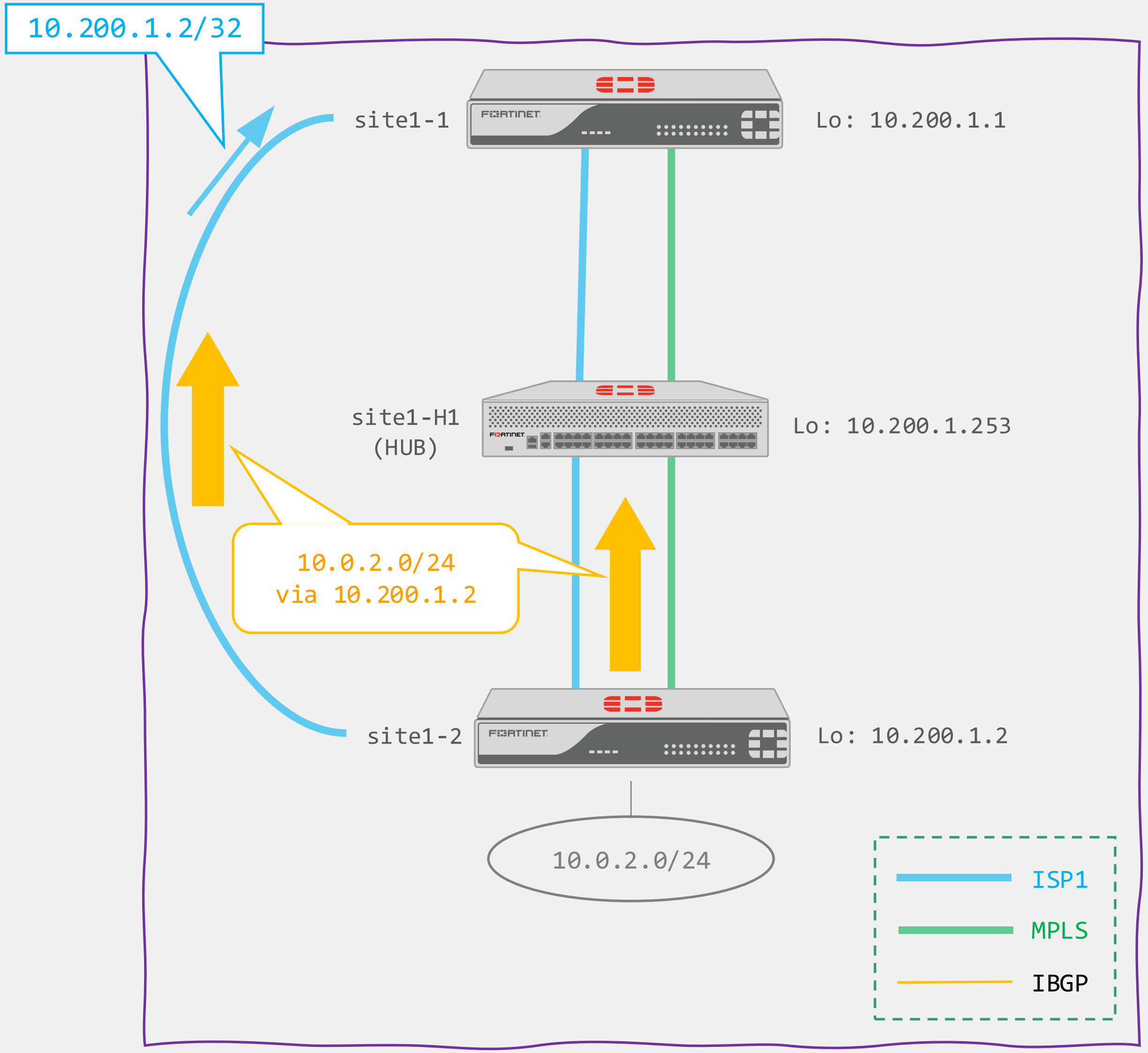
```
config router bgp
  config neighbor-group
    edit “DYN_EDGE”
      set passive disable
    next
  end
end
```



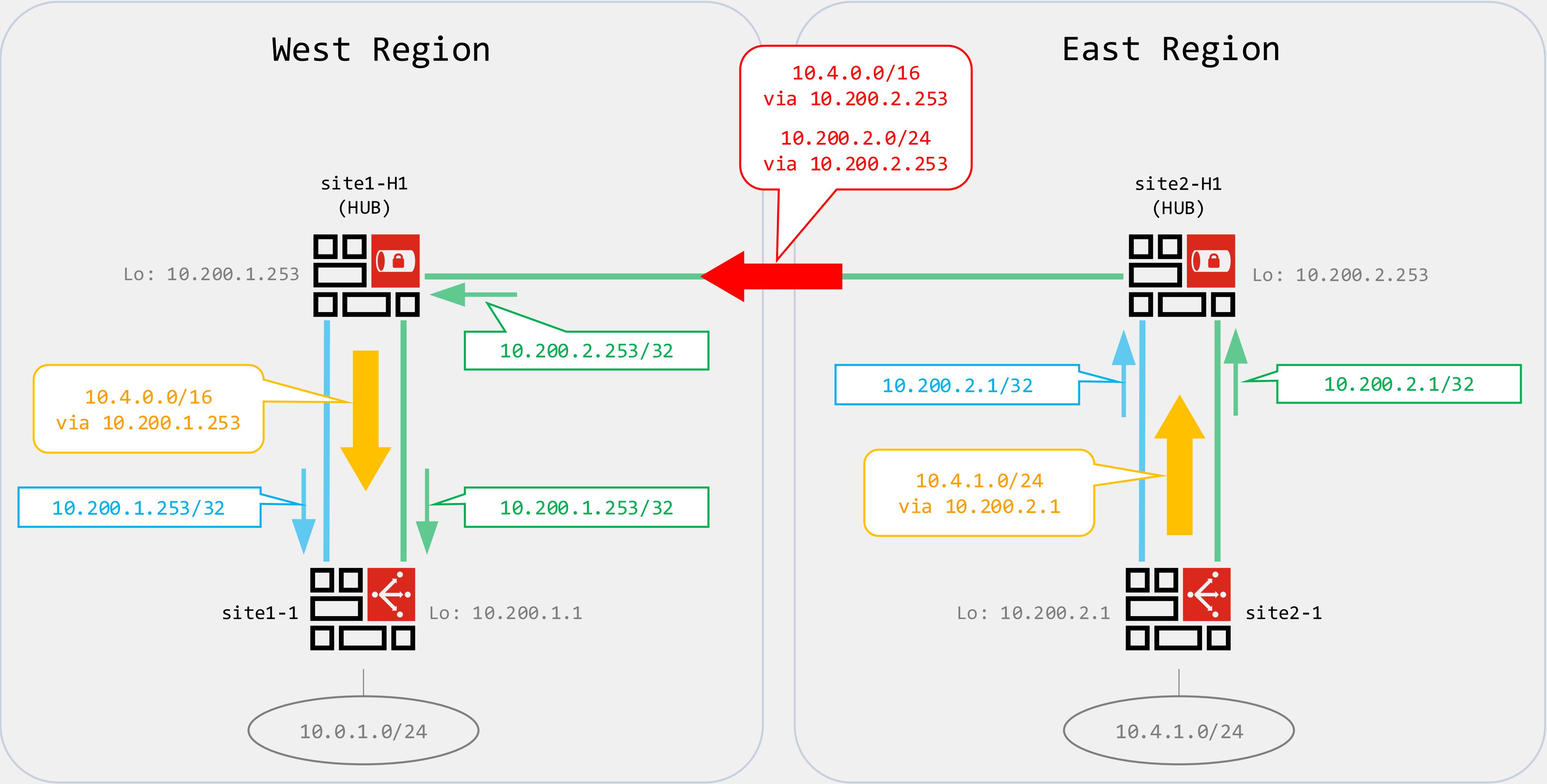
BGP on Loopback



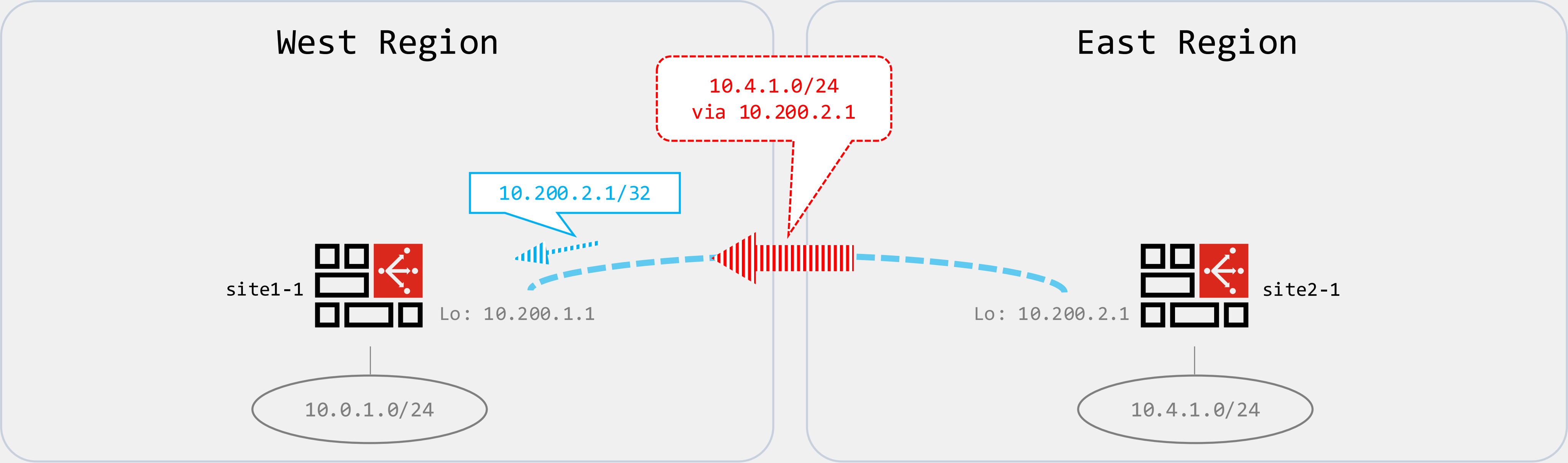
Dynamic BGP on Loopback



Horizontal Scaling



Horizontal Scaling



Multi-Regional Configuration

What is the difference between Dynamic BGP within the region and between the regions?

- Within the region, the Spoke-to-Spoke peering is **IBGP**, but between the regions it becomes **EBGP**!

Problem #1:

How to even configure it? What `remote-as` shall we put in the neighbor-group on the Spokes?

Problem #2:

By default, the routes learnt from IBGP peers are not advertised to other IBGP peers:

- A route learnt from the Hub will not be readvertised to the remote Spoke.
- A LAN prefix of the remote Spoke will not be readvertised to the Hub
- So far – that's exactly what we want!

This beautiful life breaks, when the dynamic session becomes EBGP. By default, route advertisement is allowed between IBGP and EBGP! We must restrict it!

Multi-Regional Configuration

Solution #1:

Starting from FOS 7.4.5, we can configure a **list** of remote-as!

```
config router aspath-list
  edit “SDWAN_AS”
    config rule
      edit 1
        set regexp “6500.”
        set action permit
      next
    end
  next
end
```

```
config router bgp
  config neighbor-group
    edit “DYN_EDGE”
      set remote-as-filter “SDWAN_AS”
    next
  end
end
```

Multi-Regional Configuration

Solution #2:

Tag the local LAN prefixes and make sure to advertise only them over the dynamic peering.

- Works well also if you want to add dynamic prefixes learnt from a local OSPF neighbor or what not…

```
config router bgp
  config neighbor-group
    edit “DYN_EDGE”
      set route-map-out “LAN_OUT”
    next
  end
  config network
    edit 111
      set prefix 10.0.1.0/24
      set route-map “LAN_TAG”
    next
  end
end
```

```
config router route-map
  edit “LAN_OUT”
    config rule
      edit 1
        set match-tag 100
      next
    end
  next
  edit ”LAN_TAG”
    config rule
      edit 1
        set set-tag 100
      next
    end
  next
end
```

Route Reflection is Evil?

Is it only a good thing to get rid of the RR?

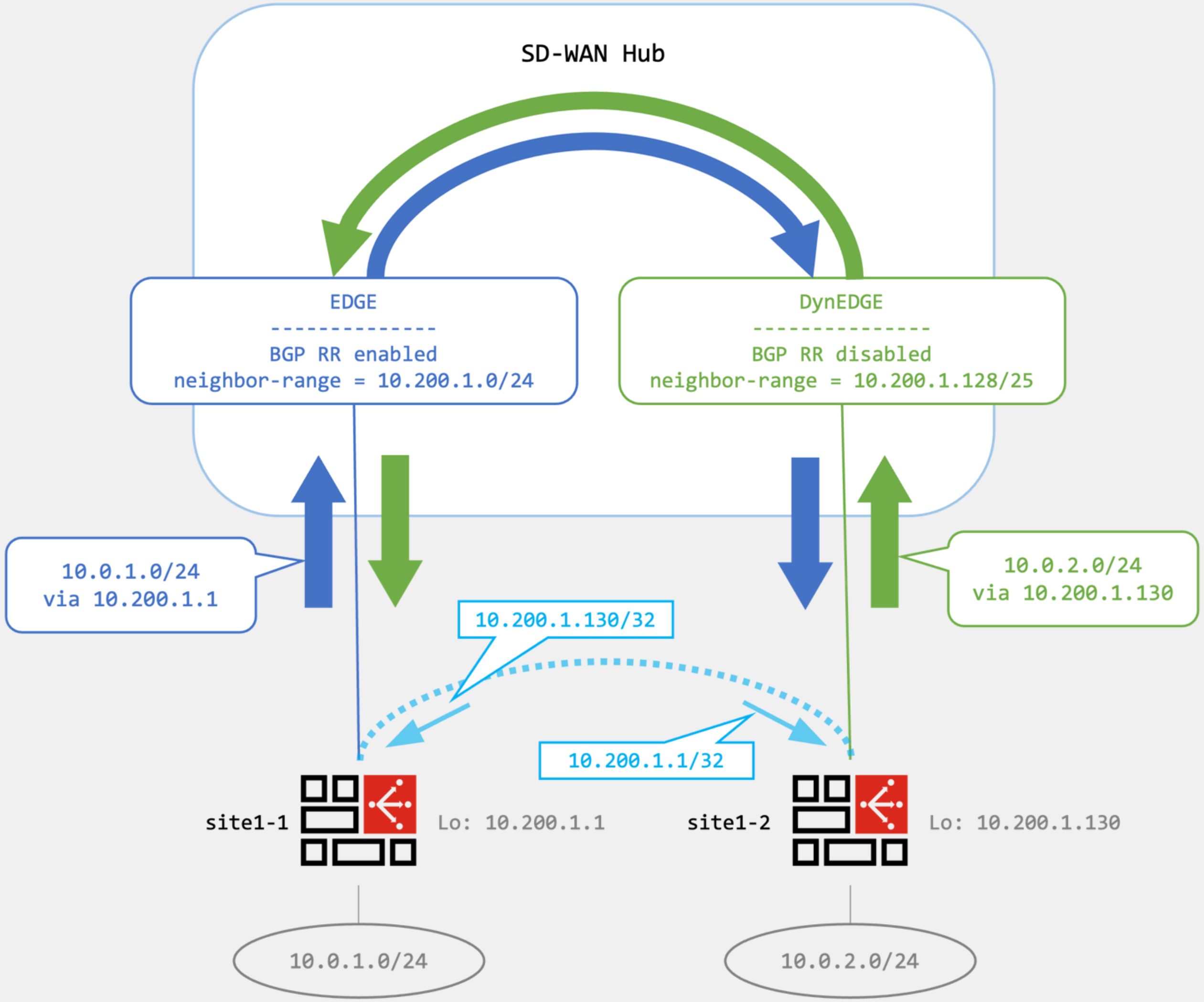
- There might be use cases where you prefer to keep the routes end-to-end
- And how about the migration? Do you have to reconfigure all your sites simultaneously?

Good news:

- Mixed RR-less + RR-based deployment works!
- You can leave the RR only where it is needed
- Recall that by default IBGP readvertises routes between RR clients and non-clients

```
config router bgp
  config neighbor-group
    edit “DynEDGE”
      set route-reflection-client disable
    next
    edit “EDGE”
      set route-reflector-client enable
    next
  end
  config neighbor-range
    edit 1
      set prefix 10.200.1.128/25
      set neighbor-group “DynEDGE”
    next
    edit 2
      set prefix 10.200.0.0/14
      set neighbor-group “EDGE”
    next
  end
end
```

Mixed Deployment



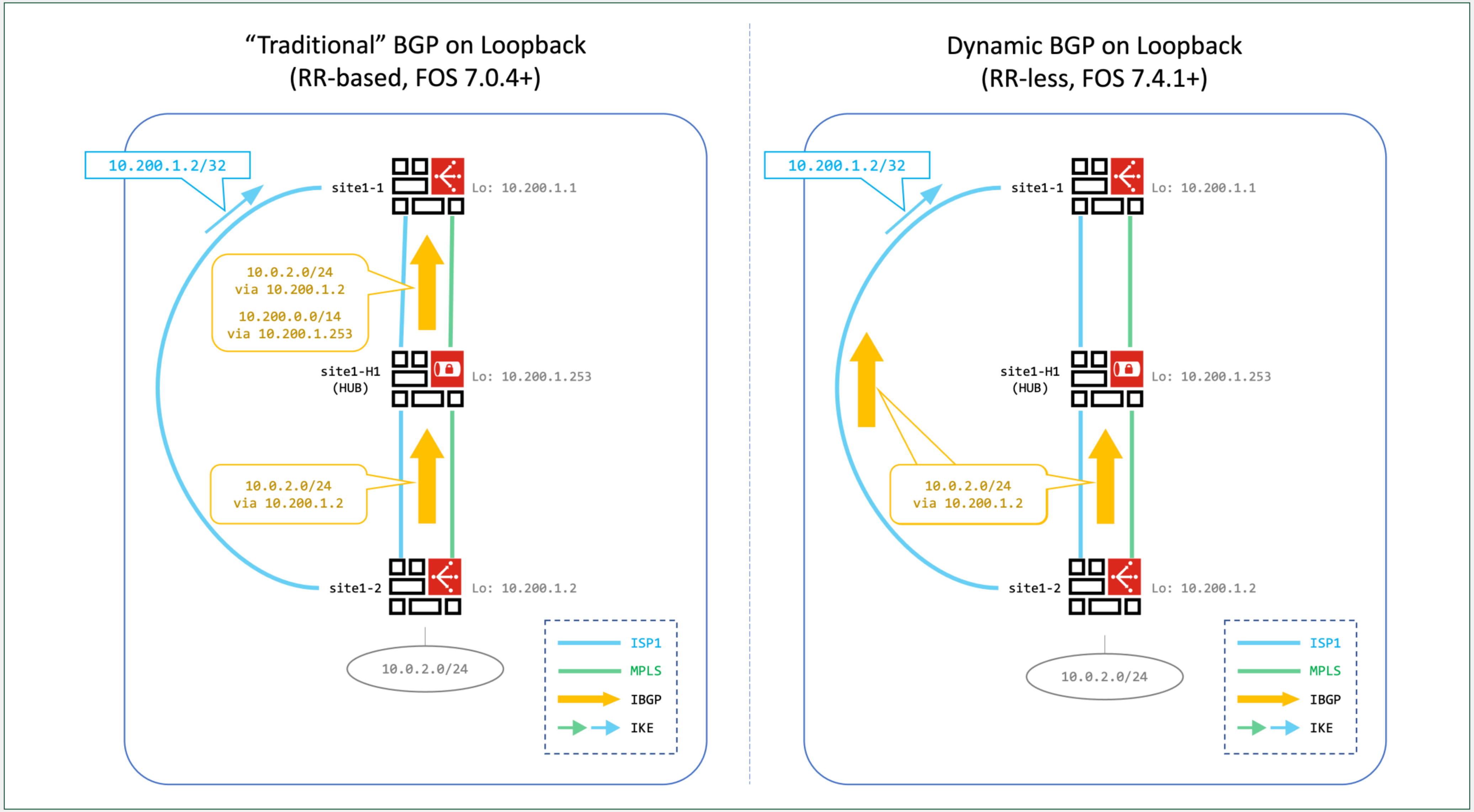
```
B  10.0.2.0/24 via 10.200.1.130 ➔ via site1-2
S  10.200.1.130/32 via site1-2
```

```
B  10.0.1.0/24 via 10.200.1.1 ➔ via site1-1
S  10.200.1.1/32 via site1-1
```

ADVPN Support Methods

RR-less vs RR-based

ADVPN Support Method



“Traditional” BGP on Loopback
(RR-based, FOS 7.0.4+)

Dynamic BGP on Loopback
(RR-less, FOS 7.4.1+)

ADVPN Support Method

Recall that we have two ADVPN support methods:

- **RR-based** – traditional, known from previous releases
- **RR-less with Dynamic BGP** – starting from FOS 7.4.1

The advantages of the RR-less Dynamic BGP design are quite straightforward:

- Scalability is greatly improved:
  - By eliminating the RR bottleneck from the Hubs
  - By allowing route summarization in multi-regional deployments
- Routing design is greatly simplified:
  - Multi-regional deployments are much easier to build
  - New hierarchical topologies become possible, virtually without constraints

No wonder that **Dynamic BGP is recommended** for most of the new deployments.

ADVPN Support Method

When should you nevertheless prefer the RR-based design?

- Existing deployments
  - No big incentive to change the routing design, if you don't hit scaling limits of the RR
- Preserving route attributes
  - Attach BGP Community to a group of Spokes and use it for steering decisions on remote Spokes
  - The routes must be reflected end-to-end, otherwise this BGP Community will be lost
- When you cannot summarize LAN subnets
  - Sometimes LAN subnets cannot be easily summarized, maybe they even use public IP ranges
  - They must be reflected, otherwise there is no way to separate them from Internet destinations

Enabling RR doesn't preclude using Dynamic BGP - you can apply RR selectively!

Multi-VRF

Updates and Tips

VRF-Aware Local-Out Traffic (7.6.1+)

Previously, local-out traffic was always sourced from VRF=0:

- This was causing different counter-intuitive effects and ”illusions”
- Local-out traffic could not match VPNv4 routes

We now support explicit `vrf-select` setting per service.

```
config system dns
  set vrf-select {{ vrf_id }}
end
config system fortiguard
  set vrf-select {{ vrf_id }}
end
config system central-management
  set vrf-select {{ vrf_id }}
end
config user ldap
  set vrf-select {{ vrf_id }}
end
config user radius
  set vrf-select {{ vrf_id }}
end
# … and so on …
```

Multicast (7.6.1+)

Previously, multicast was only supported in VRF=0:

- Both control-plane (PIM) and multicast forwarding.

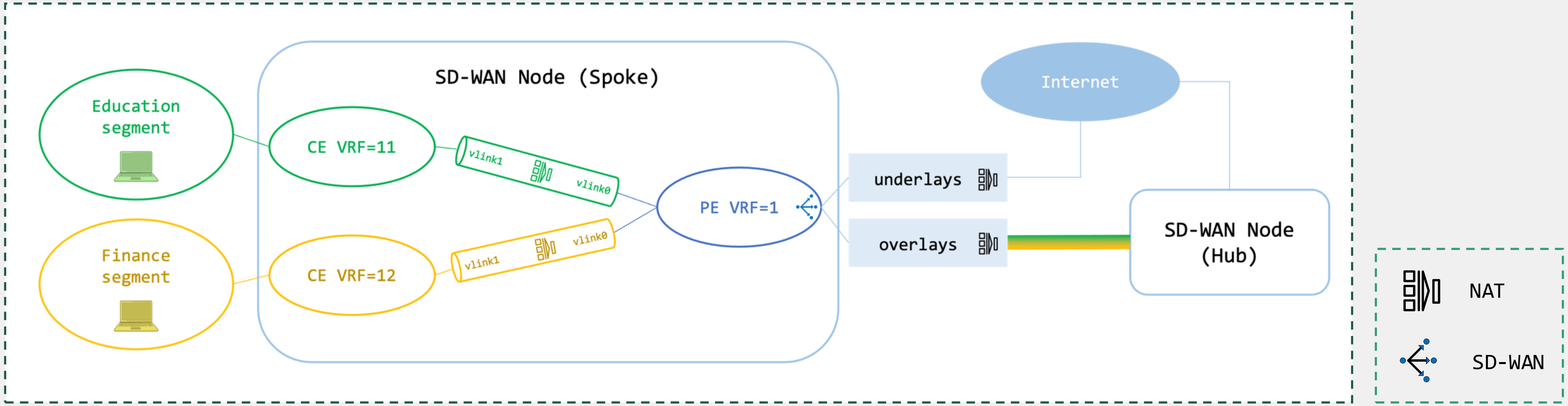
We now support multicast in a multi-VRF deployment:

- Support ”Segmentation over Single Overlay” design (`vpn-id-ipip`)
- Support VRF-aware PIM SM

Notes and Tips

- General design remains unchanged
- Starting from 7.6.1, it is no longer necessary to enable multi-VDOM mode to unhide NPU links

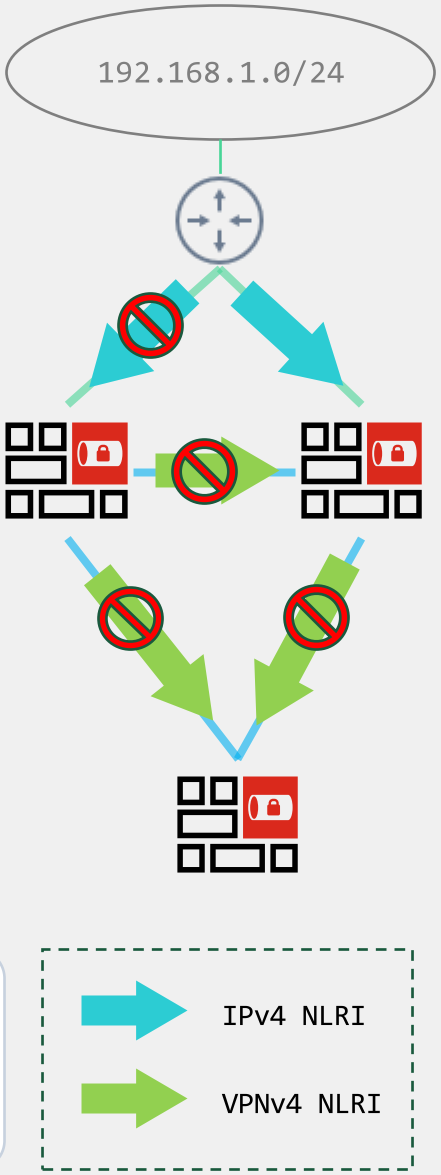
```
config system global
  set single-vdom-npuvlink enable
end
```



Notes and Tips

- The RD value should be be *unique per node and VRF* (and not just *per VRF*, as we advised before)
    - Duplicate RD on different nodes can cause MP-BGP misbehavior in multi-homing scenarios
    - Our Jinja Orchestrator is now using the Loopback to generate the RD

{{ loopback }}
RD = ~~{{ as }}~~:{{ vrf_id }}
RT = {{ as }}:{{ vrf_id }}



Notes and Tips

- The rest of our recommendations remain unchanged:
  - Avoid using VRF=0 for CE/PE
  - Prefer PE VRF = 1
  - Use CE VRFs = 2, 3, …
- Starting from FOS 7.6.1, we support 512 VRFs per VDOM (0 – 511)
  - Hence (keeping VRF=0 aside): 1 PE VRF + up to 510 CE VRFs

Remote Health Signaling

Hub-to-Edge SD-WAN

Hub-to-Edge SD-WAN

Sessions can be originated also behind Hubs (Hub-to-Spoke):

- Workloads hosted behind the Hubs
- Corporate sites outside of the SD-WAN network

Unlike Spokes, the Hubs do not actively probe the overlay health:

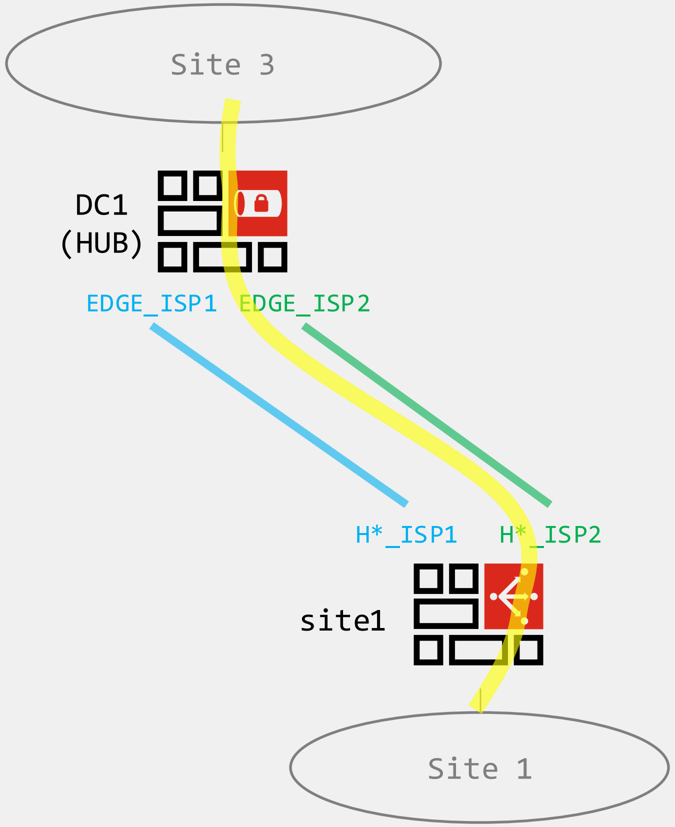
- This would be a scaling bottleneck: a single Hub can serve hundreds or thousands of Spokes
- This would be a duplicate effort: each Spoke already probes the same overlays bi-directionally

So how can the Hub steer traffic intelligently?

And how about advertising our preferences outside of the SD-WAN network?

Hub-to-Edge SD-WAN

Problem #1: How can a Hub select the best overlay?

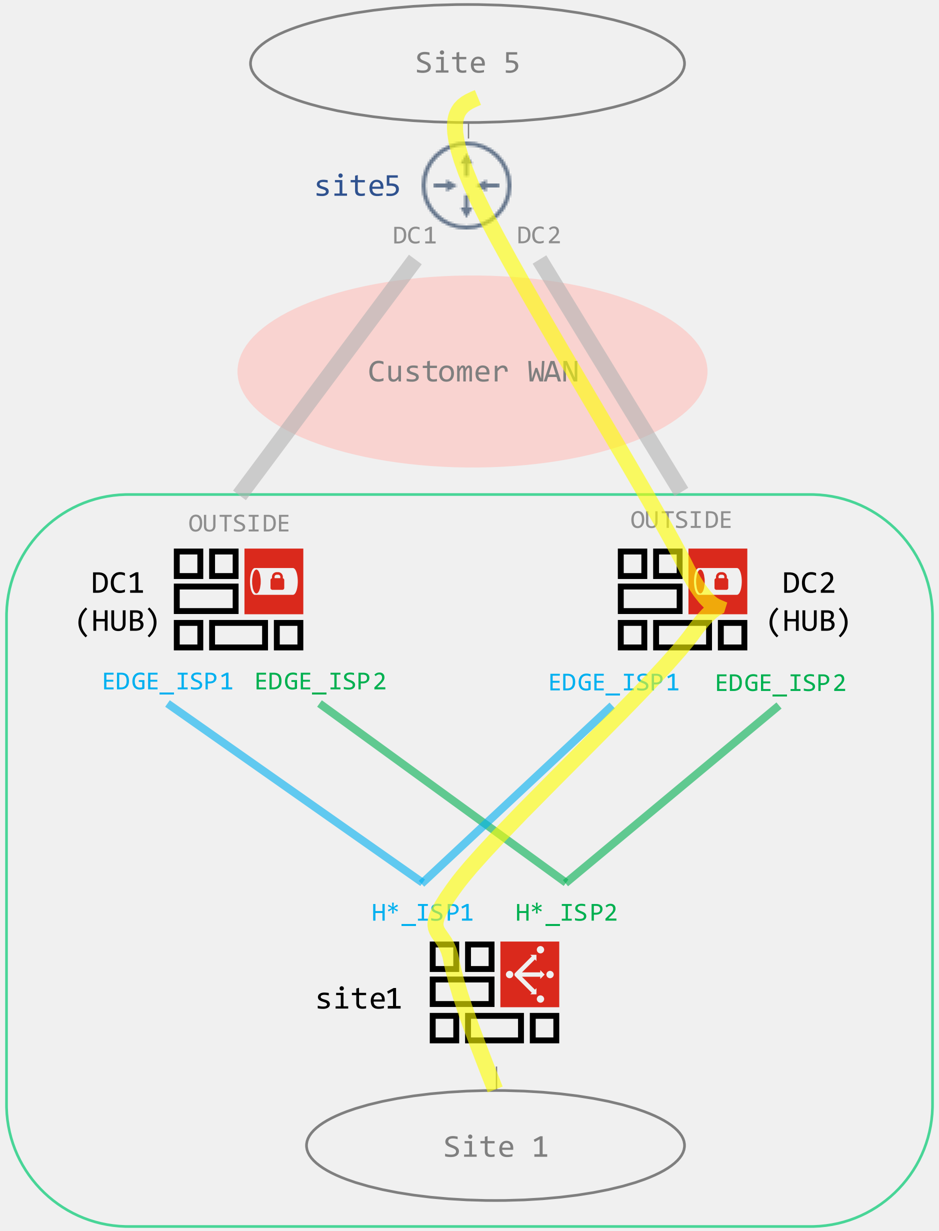


Hub-to-Edge SD-WAN

Problem #1: How can a Hub select the best overlay?

Problem #2: How can an external site select the best Hub as an entry point?

- External site may belong to another SD-WAN solution or even have no SD-WAN functionality
- 3rd-party device



History

Known by different names, such as **BGP Self-Healing** or **Hub-to-Edge SD-WAN** but we prefer to call it **Remote Health Signaling**.

	Problem #1: Hub-to-Spoke Sessions	**Problem #2: Advertising Preferred Hub**

`Release 6.4/7.0/7.4/7.6`
`BGP per Overlay`
	`SD-WAN Neighbor (per overlay) / route-map-out-preferable`	

`Release 7.0`
`BGP on Loopback`
	`No solution (with some exceptions)`	

`Release 7.2/7.4`
`BGP on Loopback`
	`embed-measured-health`
	`(Hub defines priorities)`
		`SD-WAN Neighbor (per Hub) /`
		`route-map-out-preferable`

`Release 7.6`
`BGP on Loopback`
	`embed-measured-health`
	`(Spoke defines priorities)`
		`SD-WAN Neighbor (per Hub) /`
		`route-metric`

Revamped Remote Signaling (7.6.1+)

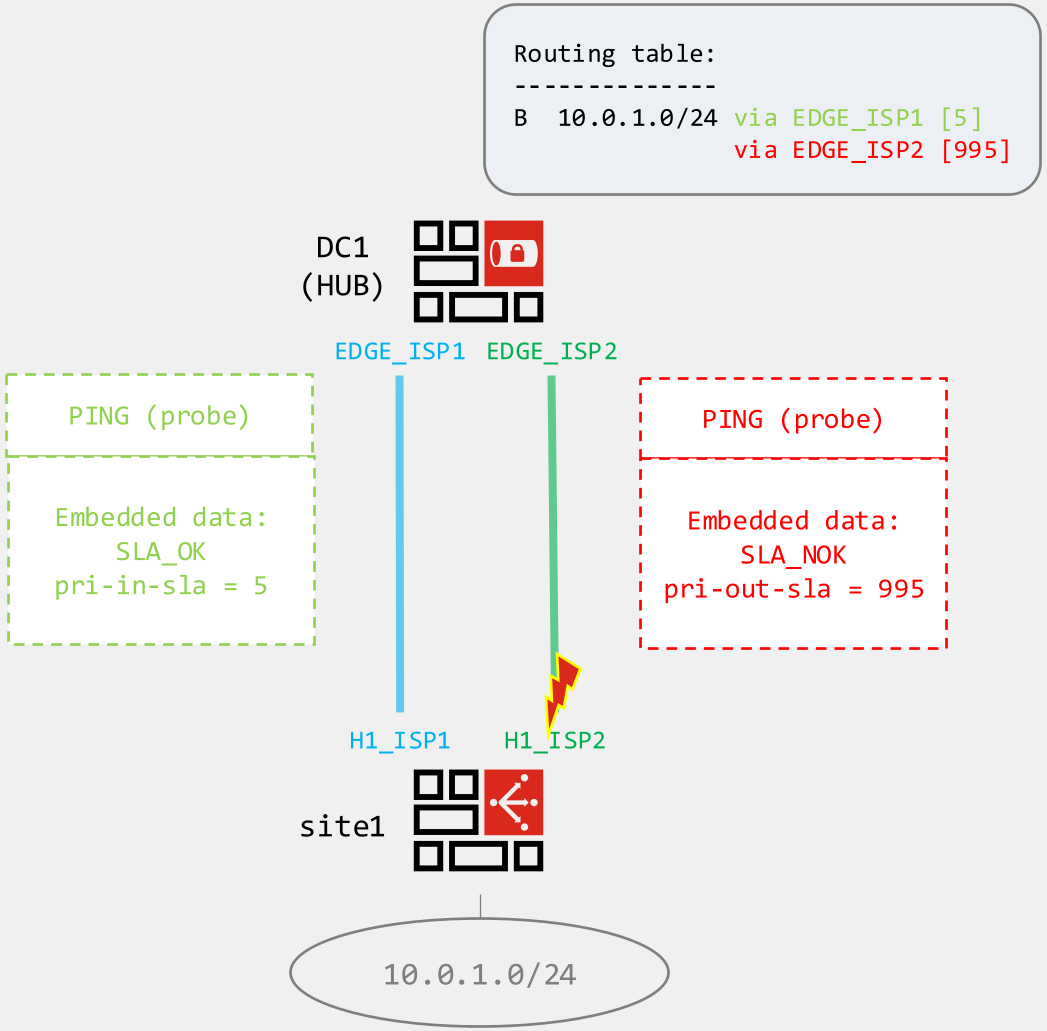
Each Spoke signals its overlay preferences:

The Spoke applies its SLA targets to each overlay member

For each member, there is a pair of route priorities configured – `priority-in-sla` and `priority-out-sla`

Once the Spoke determines the current priority for each member, it will do two things:

1. It will send these priorities to the Hub (per overlay), so that the Hub will apply them to the routes, to solve **Problem #1**.



```
Routing table:
--------------
B  10.0.1.0/24 via EDGE_ISP1 [5]
               via EDGE_ISP2 [995]
```

Revamped Remote Signaling (7.6.1+)

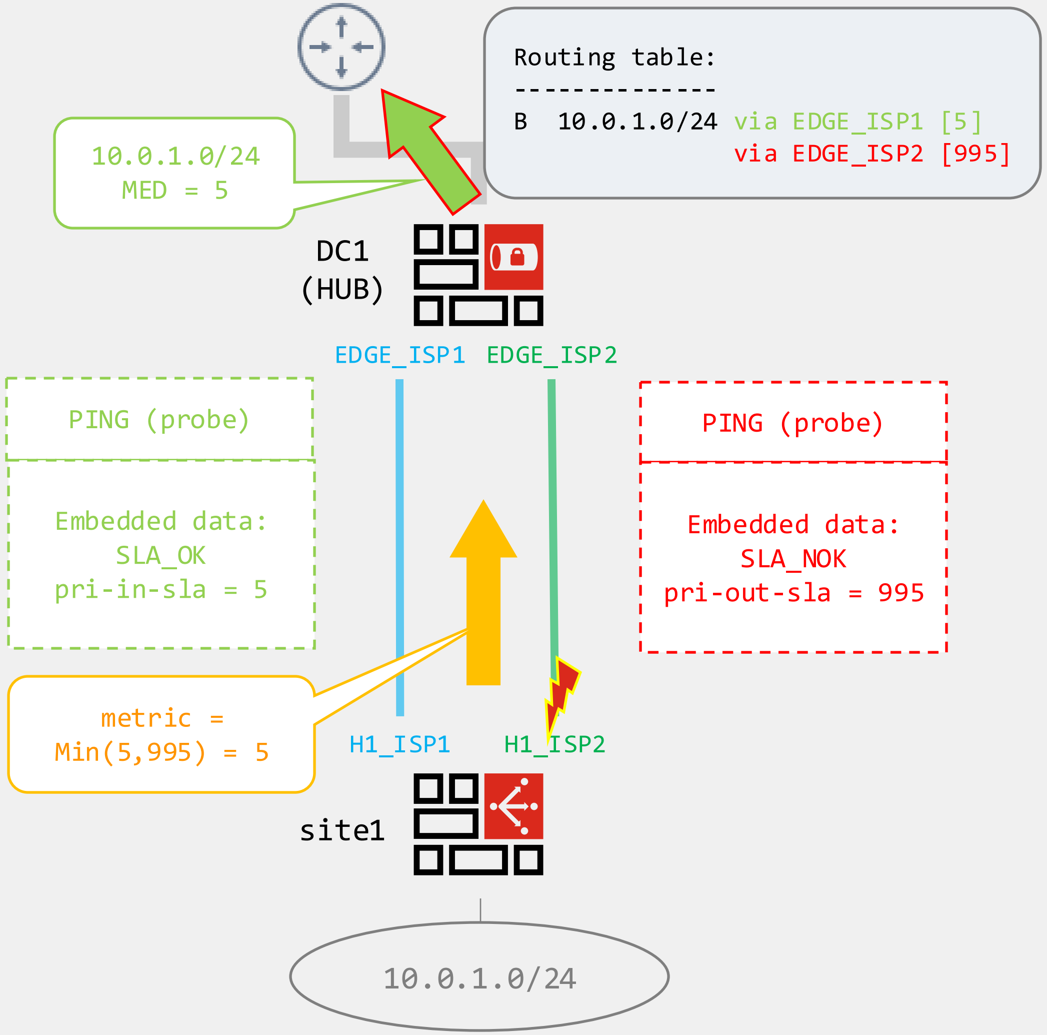
Each Spoke signals its overlay preferences:

The Spoke applies its SLA targets to each overlay member

For each member, there is a pair of route priorities configured – `priority-in-sla` and `priority-out-sla`

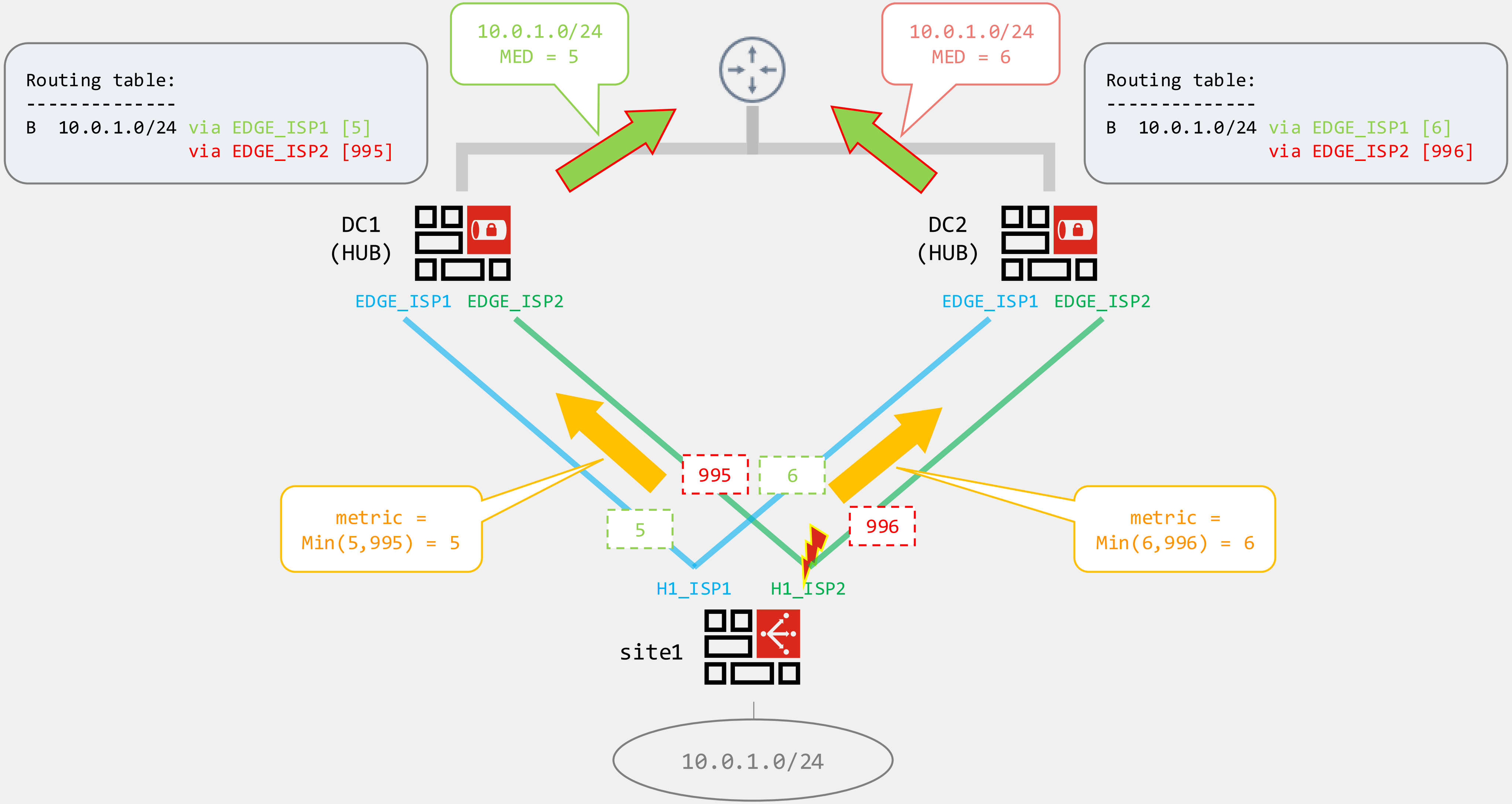
Once the Spoke determines the current priority for each member, it will do two things:

1. It will send these priorities to the Hub (per overlay), so that the Hub will apply them to the routes, to solve **Problem #1**.
2. It will send a metric (min. of all current member priorities) to the local BGP daemon, which will readvertise it upstream, to solve **Problem #2**.



```
Routing table:
--------------
B  10.0.1.0/24 via EDGE_ISP1 [5]
               via EDGE_ISP2 [995]
```

Revamped Remote Signaling (7.6.1+)



```
Routing table:
--------------
B  10.0.1.0/24 via EDGE_ISP1 [5]
               via EDGE_ISP2 [995]
```

10.0.1.0/24
MED = 5

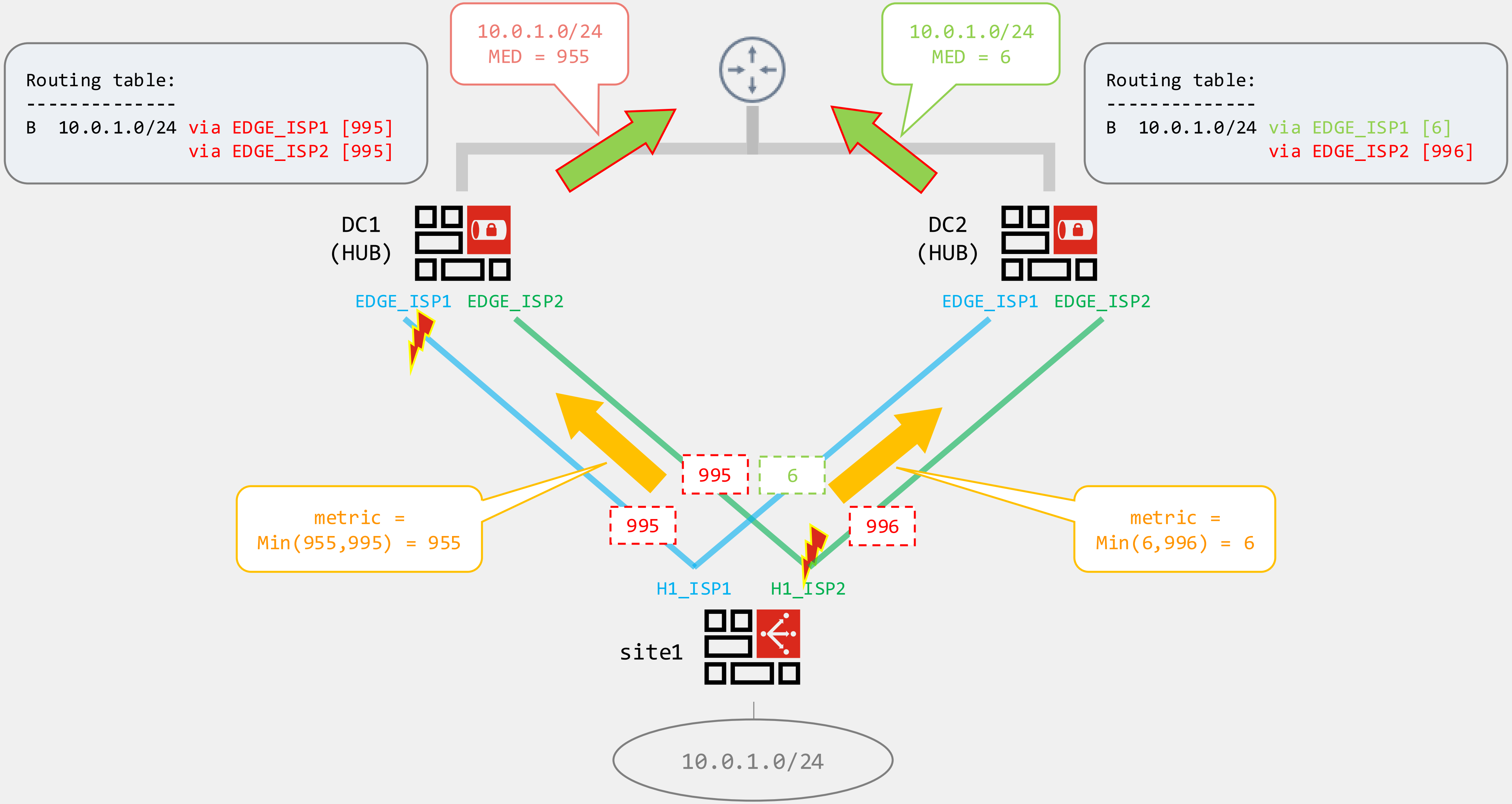
10.0.1.0/24
MED = 6

```
Routing table:
--------------
B  10.0.1.0/24 via EDGE_ISP1 [6]
               via EDGE_ISP2 [996]
```

metric =
Min(5,995) = 5

metric =
Min(6,996) = 6

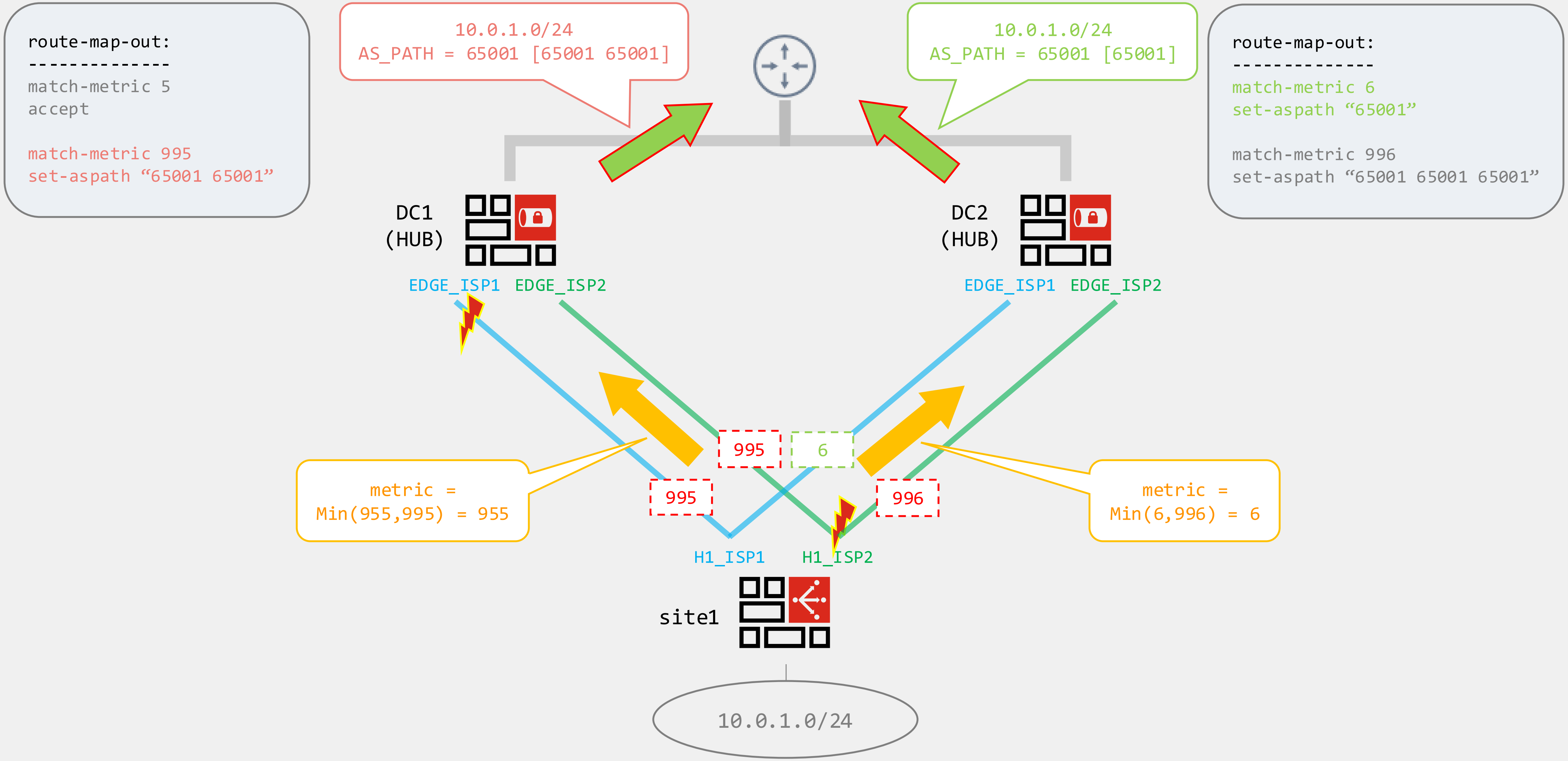
Revamped Remote Signaling (7.6.1+)



```
Routing table:
---------------
B   10.0.1.0/24 via EDGE_ISP1 [995]
                via EDGE_ISP2 [995]
```

```
Routing table:
---------------
B   10.0.1.0/24 via EDGE_ISP1 [6]
                via EDGE_ISP2 [996]
```

Revamped Remote Signaling (7.6.1+)



```
route-map-out:
--------------
match-metric 5
accept

match-metric 995
set-aspath “65001 65001”
```

```
route-map-out:
--------------
match-metric 6
set-aspath “65001”

match-metric 996
set-aspath “65001 65001 65001”
```

Revamped Remote Signaling (7.6.1+)

This design allows the Spoke to take control of the traffic flow in both directions:

- Outbound – using SD-WAN rules, with **per-application** granularity
- Inbound – using Remote Signaling, with **per-overlay** granularity

Note the flexibility: the Spoke can not only signal whether each overlay is healthy or not, but also signal its preference between the overlays (without the need for any SD-WAN configuration on the Hub):

- In our example, the Spoke wants everyone to prefer INET over MPLS (1st criteria), and then prefer Hub1 over Hub2 (2nd criteria)
- Setting in/out-sla priorities per member, we can signal any arbitrary preference for the incoming traffic.

```
config system sdwan
  config neighbor
    edit "10.200.1.253"
      set route-metric priority
      set member 3 4
    next
    edit "10.200.1.254"
      set route-metric priority
      set member 5 6
    next
  end
end
```

```
config system sdwan
  config members
    edit 3
      set interface “H1_INET”
      set priority-in-sla 5
      set priority-out-sla 995
    next
    edit 4
      set interface “H1_MPLS”
      set priority-in-sla 8
      set priority-out-sla 998
    next
    edit 5
      set interface “H2_INET”
      set priority-in-sla 6
      set priority-out-sla 996
    next
    edit 6
      set interface “H2_MPLS”
      set priority-in-sla 9
      set priority-out-sla 999
    next
  end
end
```

FORTINET®