Table of Contents

Introduction 1
MP-BGP Overview 1
VPNv4 Prefixes and EIGRP Extended Communities 3
VPNv4 Prefixes and Redistribution 4
Race Condition 1: Backdoor Link preferred by EIGRP 8
BGP Cost Community 10
Race Condition 2: EIGRP Routing Loops 12
Deploying EIGRP SoO 14
Conclusions 19

Introduction

Probably one of the most confusing topics for people studying BGP/MPLS VPNs are EIGRP SoO and Cost communities. Not the concepts themselves, but rather their purpose. Lack of clear documentation makes it hard to understand why and how some of the specific features are implemented. In this blog post we discuss the BGP Cost community and EIGRP SoO attribute. First, a brief overview of MP-BGP is given along with general concepts of route redistribution. Next, the problems that arise when using EIGRP in multihomed VPN environment are demonstrated. Lastly, two specific EIGRP features are described and their use demonstrated. As usual, a reader is assumed to have basic understanding of L3 VPNs and PE-CE routing concepts.

MP-BGP Overview

The core of MPLS VPN functionality is not the MPLS technology but Multiprotocol BGP extension. Any tunneling solution could be used in place of MPLS. Being the true workhorse, MP-BGP allows BGP for carrying any “routable” protocol’s reachability information in BGP. Best path selection is still performed based on the BGP attributes, only for the matching “multiprotocol prefixes”. To better understand how MP-BGP works, start by recalling the classic BGP (RFC1771) UPDATE message format. It consists of the following sections:

UPDATE = [Withdrawn prefixes (Optional)] + [Path Attributes] + [NLRIs].

The Withdrawn Prefixes and NLRIs are formatted as *IPv4 prefixes*, and their structure does not support any other routable protocols. The Path Attributes (e.g. AS_PATH, ORIGIN, LOCAL_PREF, and NEXT_HOP) are associated with all NLRIs packed in the update; prefixes sharing different set of path attributes should be carried in a separate UPDATE message. Also, notice that the NEXT_HOP is a BGP attribute, and it contains an IPv4 address as well.

In order to introduce support for non-IPv4 network protocols into BGP, two new optional transitive *Path Attributes* have been added to BGP. Notice that this does not affect the original UPDATE packet formatting at all. The first attribute is known as MP_REACH_NLRI, has the following structure:

MP UPDATE = [AFI/SAFI] + [NEXT_HOP] + [NLRI].

Here NLRI stands for *Network Layer Reachability Information*. In this packet, both NEXT_HOP and NLRI are formatted according to the protocol encoded via the AFI/SAFI pair. This abbreviation stands for *Address Family Identifier* and *Subsequent Address Family Identifier* respectively. For example, this could be an IPv6 or CLNS prefix. Thus, all information about non-IPv4 prefixes is encoded in a new BGP Path Attribute. A typical MP BGP UPDATE message that contains MP_REACH_NLRI attributes would have no “classic” IPv4 NEXT_HOP attribute and no “Withdrawn Prefixes” or “NLRIs” found in normal UPDATE messages – all information is conveyed in BGP attributes. For the next-hop calculations, the receiving BGP speaker should use the information found in MP_REACH_NLRI attribute. However, the multi-protocol UPDATE message may contain other BGP path attributes such as AS_PATH, ORIGIN, MED, LOCAL_PREF and so on to allow the BGP speakers performing classic best-path selection. However, these attributes are associated with the non-IPv4 prefixes found in all attached MP_REACH_NLRI attributes.

The second attribute, MP_UNREACH_NLRI has format similar to MP_REACH_NLRI but lists the “multi-protocol” addresses to be removed – it services function similar to the BGP Withdrawn Prefixes field. No other path attributes need to be sent with this attribute, an UPDATE message may simply contain the list of MP_UNREACH_NLRIs.

The list of supported AFIs may be found in RFC1700 (though it’s obsolete now, it is still very informative). For example, AFI 1 stands for IPv4, AFI 2 stands for IPv6 etc. The subsequent AFI is needed to clarify the purpose of the information found in MP_REACH_NLRI. For example, SAFI value of 1 means the prefixes should be used for *unicast* forwarding, SAFI 2 means the prefixes are to be used for multicast RPF checks and SAFI 3 means the prefixes could be used for both purposes. Last, but not least – SAFI of 128 means MPLS labeled VPN address.

Just as a reminder, BGP process would perform separate best-path election process for “classic” IPv4 prefixes and every AFI/SAFI pair prefixes separately, based on the respective path attributes. This allows for independent route propagation for the addresses found in different families. Since a given BGP speaker may not support particular network protocols, the list of supported AFI/SAFI pairs is advertised using BGP capabilities feature (another BGP extension), and the particular network protocol information is only propagated if both speakers support it.

VPNv4 Prefixes and EIGRP Extended Communities

VPNv4 prefixes form the core of BGP VPNs. As mentioned above, VPNv4 prefixes are identified by the AFI/SAFI pair of 1/128. The MP NLRI has the format (roughly) **[RD:VPN IPv4 Prefix:Mux Label]** – the label here services multiplexing purposes to allow for proper VRF selection at the egress PE. Since L3 VPNs normally connect discrepant sites using the same IGP routing protocol, a way to carry the IGP information transparently was required. Using just BGP MED alone was not enough to propagate this information: for example, the composite EIGRP metric components as well as EIGRP AS number would be missing. Based on this, various extended BGP communities were routinely created to convey this additional information. Extended communities are very similar to regular communities: they are 64-bit quantities that have a type field (2 bytes) and arbitrary numeric content following the type. It's up to the receiving BGP speaker to interpret the extended-community values. Special community types have been allocated for EIGRP (taken from http://www.cisco.com/en/US/tech/tk436/tk428/technologies_tech_note09186a00801eb09a.shtml)

EIGRP Attribute	Type	Usage	Value
General	0x8800	EIGRP General Route Information	Route Flag and Tag
Metric	0x8801	EIGRP Route Metric Information and Autonomous System	Autonomous System and Delay
	0x8802	EIGRP Route Metric Information	Reliability, Next Hop, and Bandwidth
	0x8803	EIGRP Route Metric Information	Reserve, Load and Maximum Transmission Unit (MTU)
External	0x8804	EIGRP External Route Information	Remote Autonomous System and Remote ID
	0x8805	EIGRP External Route Information	Remote Protocol and Remote Metric

.

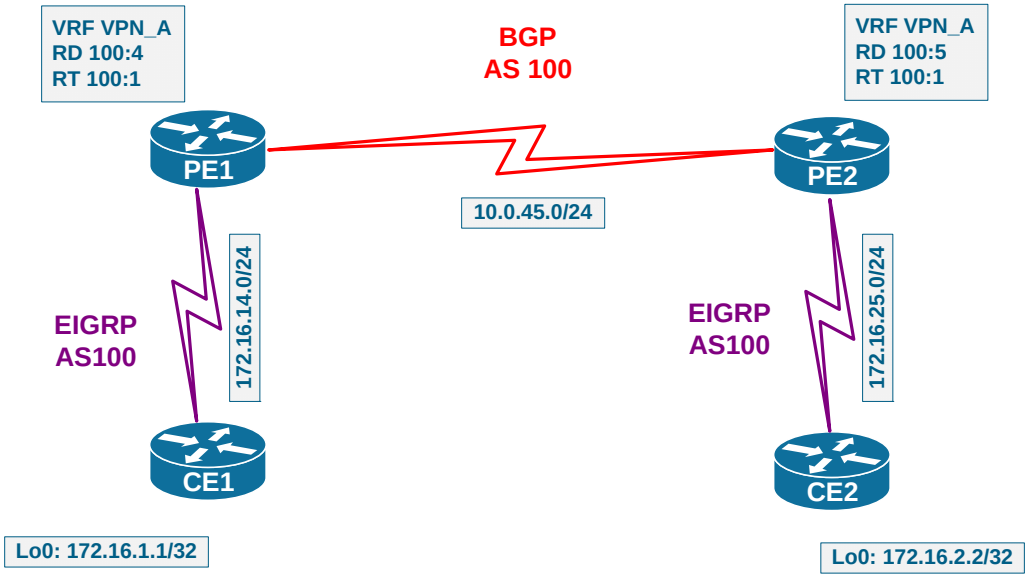
As you can see, these attributes could be used to carry all EIGRP extended metric information, including the information on external route source. Notice that all community types start with 0x88 – this might be due to the fact that EIGRP's IP protocol number is 88 (though decimal). Here is a sample output from BGP table:


```
PE1#show bgp vpnv4 unicast all 172.16.2.2
BGP routing table entry for 100:4:172.16.2.2/32, version 18
Paths: (1 available, best #1, table VPN_A)
  Not advertised to any peer
  Local, imported path from 100:5:172.16.2.2/32
    10.0.5.5 (metric 65) from 10.0.5.5 (10.0.5.5)
      Origin incomplete, metric 2688000, localpref 100, valid, internal, best
      Extended Community: RT:100:1
        Cost:pre-bestpath:128:2688000 (default-2144795647) 0x8800:32768:0
        0x8801:100:2688000 0x8802:65281:1657856 0x8803:65281:1500
      mpls labels in/out nolabel/23
```

Let's see how MP-BGP and EIGRP interact together by means of redistribution process.

VPNv4 Prefixes and Redistribution

Look at the sample topology below.



Here PE1 and PE2 redistribute between EIGRP AS 100 process for VRF VPN_A and BGP process for AS 100. The redistribution process is slightly different than it would have been if no VRFs were present. In short, this is what happens when CE2 advertises the prefix 172.16.2.2/32 into EIGRP

1. PE2 receives the EIGRP update and redistributes the prefix into BGP. During this process, EIGRP specific metrics are stored as BGP extended communities. PE2 sends the MP-BGP update with RT 100:1, RD 100:5 and extended communities to PE1.
2. The MP-BGP update reaches PE1 and enters the VPNv4 routing table. Before this route could be redistributes into the respective VRF, the BGP import process needs to run in order to make the prefixes comparable (change the RDs). The BGP import process runs every 15 seconds by default, and this timer could be adjusted under the vpnv4 address-family configuration using the command **`bgp scan-time import`**. As usual, the import process uses the RT values to properly redistribute the prefixes. Notice that BGP withdraws are processed immediately, bypassing the import process, to improve failure recovery.
3. During the import process, PE2's BGP prefix for 172.16.2.2/32 will be re-injected into BGP with the RD value of 100:4. At this moment, it is possible to elect a single best path based on BGP best-path selection rules. The best path is further injected into VRF VPN_A's EIGRP process. Redistribution process takes in account the extended communities and reconstructs the EIGRP metric. PE2's prefix 172.16.2.2/32 is now seen by the EIGRP process as internal EIGRP route with the same metric it had on R5. Essentially, the SP cloud is interpreted as a link with EIGRP metric of zero, and PE1 sees the new prefix as "directly connected" with the reconstructed metric.
4. EIGRP selects the best path based on shortest EIGRP metric and install it into the VRF routing table. If the best path was learned from BGP, the route is installed as BGP route, *not* as an EIGRP one. This prevents local redistribution loops between EIGRP and BGP. If the EIGRP process selects non-VPNv4 prefix as the best one, it installs a native EIGRP prefix into the VRF routing table.

The BGP extended communities carry the information of the EIGRP AS number of the redistributing router. The importing EIGRP process only considers the BGP-learned routes internal if the source AS (found in BGP extended communities) equals the local AS. Otherwise, the prefixes are treated as if they were redistributed from another EIGRP process: metrics are preserved but prefixes are marked as external. This feature allows for supporting proper inter-VPN route exchange with multiple sites using EIGRP as the routing protocol. Here are sample show commands output for the loop less EIGRP VPN design, based on the topology provided above:

```
CE1#show ip route eigrp
     172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
D       172.16.25.0/24 [90/5120000] via 172.16.14.4, 00:00:09, Serial0/0.14
D       172.16.2.2/32 [90/5248000] via 172.16.14.4, 00:00:09, Serial0/0.14

CE2#sh ip route eigrp
     172.16.0.0/16 is variably subnetted, 4 subnets, 2 masks
D       172.16.14.0/24 [90/5120000] via 172.16.25.5, 00:03:32, Serial0/0.25
D       172.16.1.1/32 [90/5248000] via 172.16.25.5, 00:00:03, Serial0/0.25
```

Notice that the topology entries show “VPNv4 Sourced” and have the advertised metric value of zero.

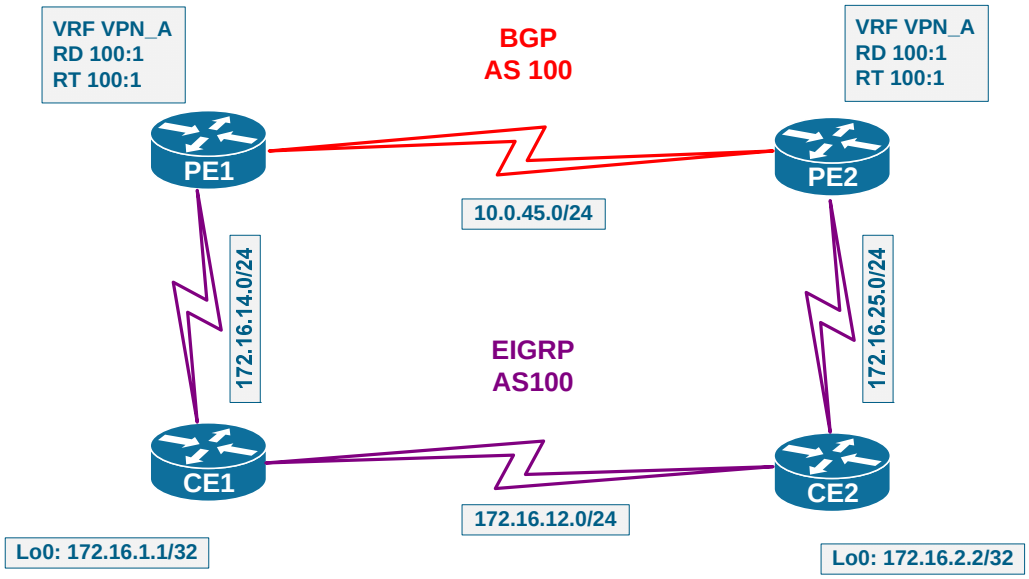
```
PE1#show ip eigrp vrf VPN_A topology
IP-EIGRP Topology Table for AS(100)/ID(172.16.14.4) Routing Table: VPN_A

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 172.16.25.0/24, 1 successors, FD is 2560000
        via VPNv4 Sourced (2560000/0)
P 172.16.14.0/24, 1 successors, FD is 2560000
        via Connected, Serial0/0/0.14
P 172.16.1.1/32, 1 successors, FD is 2688000
        via 172.16.14.1 (2688000/128000), Serial0/0/0.14
P 172.16.2.2/32, 1 successors, FD is 2688000
        via VPNv4 Sourced (2688000/0)

PE2#show ip eigrp vrf VPN_A topology
IP-EIGRP Topology Table for AS(100)/ID(172.16.15.5) Routing Table: VPN_A

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 172.16.25.0/24, 1 successors, FD is 2560000
        via Connected, Serial0/0/0.25
P 172.16.14.0/24, 1 successors, FD is 2560000
        via VPNv4 Sourced (2560000/0)
P 172.16.1.1/32, 1 successors, FD is 2688000
        via VPNv4 Sourced (2688000/0)
P 172.16.2.2/32, 1 successors, FD is 2688000
        via 172.16.25.2 (2688000/128000), Serial0/0/0.25
```

Now look at the corresponding BGP tables for PE1 and PE2:

```
PE1#show bgp vpnv4 unicast all
BGP table version is 299, local router ID is 10.0.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN_A)
*> 172.16.1.1/32    172.16.14.1        2688000         32768 ?
*>i172.16.2.2/32    10.0.5.5           2688000    100      0 ?
*> 172.16.14.0/24   0.0.0.0                  0         32768 ?
*>i172.16.25.0/24   10.0.5.5                 0    100      0 ?

PE1#show bgp vpnv4 unicast all 172.16.1.1
BGP routing table entry for 100:1:172.16.1.1/32, version 299
Paths: (1 available, best #1, table VPN_A)
  Advertised to update-groups:
        1
  Local
    172.16.14.1 from 0.0.0.0 (10.0.4.4)
      Origin incomplete, metric 2688000, localpref 100, weight 32768, valid,
sourced, best
      Extended Community: RT:100:1
        Cost:pre-bestpath:128:2688000 (default-2144795647) 0x8800:32768:0
        0x8801:100:2688000 0x8802:65281:1657856 0x8803:65281:1500
      mpls labels in/out 29/nolabel

PE2#show bgp vpnv4 unicast all
BGP table version is 35, local router ID is 10.0.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN_A)
*>i172.16.1.1/32    10.0.4.4           2688000    100      0 ?
*> 172.16.2.2/32    172.16.25.2        2688000         32768 ?
*>i172.16.14.0/24   10.0.4.4                 0    100      0 ?
*> 172.16.25.0/24   0.0.0.0                  0         32768 ?

PE2#show bgp vpnv4 unicast all 172.16.1.1
BGP routing table entry for 100:1:172.16.1.1/32, version 35
Paths: (1 available, best #1, table VPN_A)
Flag: 0x820
  Not advertised to any peer
  Local
    10.0.4.4 (metric 65) from 10.0.4.4 (10.0.4.4)
      Origin incomplete, metric 2688000, localpref 100, valid, internal, best
      Extended Community: RT:100:1
        Cost:pre-bestpath:128:2688000 (default-2144795647) 0x8800:32768:0
        0x8801:100:2688000 0x8802:65281:1657856 0x8803:65281:1500
      mpls labels in/out nolabel/29
```

Race Condition 1: Backdoor Link preferred by EIGRP

In the case where all sites have single point of attachment to the SP core there are no redistribution loops. However, if you have a site with backdoor link, EIGRP information may theoretically circulate in the loop as there is no automatic way to detect this. Look at the multihomed topology below, which employs redistribution between MP-BGP and EIGRP at two different points (PE1/PE2):



There could be certain race conditions in the above scenario, based on the timing of EIGRP/BGP updates. Race conditions are not uncommon in gradient algorithms, which accept best information instantly, based solely on the local decision. This behavior is non-deterministic may lead to some unwanted or even dangerous behavior.

Imagine that CE2's prefix 172.16.2.2/32 reaches PE1 via MP-BGP first. If prefix is imported into VRF without any delay, EIGRP will accept it and interpret as an internal prefix. If then the same prefix reaches PE1 across the backdoor path (CE2->CE1->PE1) and has worse EIGRP metric compared to the VPNv4-learned prefix then EIGRP will keep the old (BGP) path.

However, and most likely, if the EIGRP update reaches PE1 before the VPNv4 route, the path will be injected into MP-BGP table on PE1. When the VPNv4 update comes in from PE2, there is already a path in the BGP table. The locally-sourced route will always be elected based on BGP's rules (prefer locally originated routes, plus the fact that IOS BGP assigns weight of 32768 to the local prefixes). This will prevent PE1 from ever selecting a potentially better path via PE2. There is a solution to this problem, known as BGP Cost community, but before we move to it, look at the snapshot of the backdoor topology that does not support the Cost community. You need to use the command **`bgp bestpath cost-community ignore`** to accomplish the same results.


```
PE1#show bgp vpnv4 unicast all
BGP table version is 312, local router ID is 10.0.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN_A)
* i172.16.1.1/32    10.0.5.5           2688256    100      0 ?
*>                  172.16.14.1        2688000             32768 ?
* i172.16.2.2/32    10.0.5.5           2688000    100      0 ?
*>                  172.16.14.1        2688256             32768 ?
* i172.16.12.0/24   10.0.5.5           2560256    100      0 ?
*>                  172.16.14.1        2560256             32768 ?
* i172.16.14.0/24   10.0.5.5           5120256    100      0 ?
*>                  0.0.0.0                  0             32768 ?
* i172.16.25.0/24   10.0.5.5                  0    100      0 ?
*>                  172.16.14.1        5120256             32768 ?

PE2#show bgp vpnv4 unicast all
BGP table version is 11, local router ID is 10.0.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN_A)
* i172.16.1.1/32    10.0.4.4           2688000    100      0 ?
*>                  172.16.25.2        2688256             32768 ?
* i172.16.2.2/32    10.0.4.4           2688256    100      0 ?
*>                  172.16.25.2        2688000             32768 ?
* i172.16.12.0/24   10.0.4.4           2560256    100      0 ?
*>                  172.16.25.2        2560256             32768 ?
* i172.16.14.0/24   10.0.4.4                  0    100      0 ?
*>                  172.16.25.2        5120256             32768 ?
* i172.16.25.0/24   10.0.4.4           5120256    100      0 ?
*>                  0.0.0.0                  0             32768 ?
```

The best paths elected by the BGP speakers above are always the local prefixes, even though the BGP-learned paths have better EIGRP metrics. Here is how the BGP cost community solves this issue.

BGP Cost Community

The idea behind the Cost community is having some additive metric, similar to IGP's costs, that could be accounted in BGP best-path selection process. This community also specifies the POI – point of insertion in the BGP best-path selection procedure. Thus, theoretically a BGP speaker may be instructed to compare cost communities say before the AS_PATH comparison or even before the weights are considered. Effectively, cost community may reprogram the best-path selection process and enforce "IGP-like" selection rules. A drawback of this design could be the fact that BGP never was designed to be "IGP-like" and effectively the use of the Cost community assumes that the cost of traversing the BGP clouds is zero. Here is the format for the BGP Cost extended-community:

[POI (1 bytes)] + [Community-ID (1 byte)] + [Cost (4 bytes)]

The use of this community is not standard, as the original draft draft-retana-bgp-custom-decision-00.txt never became and RFC. The most useful POI so far has the value of 128 – known as "ABSOLUTE" POI, it say the Cost value should be considered before the BGP best-path process starts. If a route has minimal Cost, it is immediately elected as the best path, without considering any other options. Effectively, BGP routers make decision based solely on the remote-sites metric values, which could be considered an extreme variant of "hot potato" routing. Of course, the use of the BGP Cost community provides immediate solution for the above problem with EIGRP path selection: BGP process will consider the Cost community for both local and remote paths before even comparing their weights or other preferences. And if the remotely learned path has better cost it will be elected as the best locally as well. Take a look at the same multi-homed scenario with BGP cost community enabled (**no bgp bestpath cost-community ignore**)

```
PE1#show bgp vpnv4 unicast all
BGP table version is 316, local router ID is 10.0.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN_A)
*> 172.16.1.1/32    172.16.14.1        2688000         32768 ?
*>i172.16.2.2/32    10.0.5.5           2688000    100      0 ?
* i172.16.12.0/24   10.0.5.5           2560256    100      0 ?
*>                  172.16.14.1        2560256         32768 ?
*> 172.16.14.0/24   0.0.0.0                  0         32768 ?
*>i172.16.25.0/24   10.0.5.5                 0    100      0 ?
```

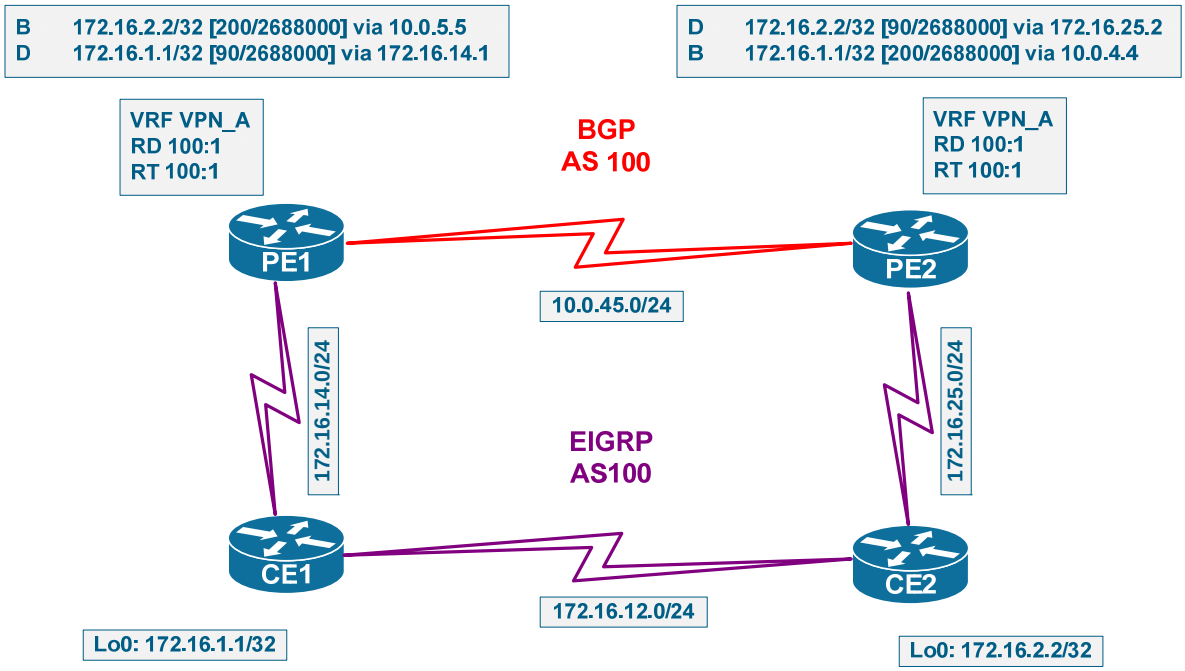


```
PE2#show bgp vpnv4 unicast all
BGP table version is 18, local router ID is 10.0.5.5
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:1 (default for vrf VPN_A)
*>i172.16.1.1/32    10.0.4.4           2688000    100      0 ?
*> 172.16.2.2/32    172.16.25.2        2688000         32768 ?
* i172.16.12.0/24   10.0.4.4           2560256    100      0 ?
*>                  172.16.25.2        2560256         32768 ?
*>i172.16.14.0/24   10.0.4.4                 0    100      0 ?
*> 172.16.25.0/24   0.0.0.0                  0         32768 ?
```

Notice that now the best-paths are the routes learned from the remote PE router. It is also important to understand why there are no “secondary” paths in the BGP table anymore. Since the remote prefixes are elected as best, they get redistributed into EIGRP and the latter selects them as optimal routes. The local EIGRP process then removes the local EIGRP prefixes from the routing table and this stop them from propagating into BGP. Thus, the BGP table ends up with just one best route.

Race Condition 2: EIGRP Routing Loops

The second issue is more dangerous. Imagine first, that we have PE1 and PE2 converged in the above-shown backdoor topology. PE1 prefers reaching CE1's Loopback0 prefix 172.16.1.1/32 directly via CE1, while PE2 prefers reaching the same prefix via the MP-BGP path, learned from PE1. This means both PEs are configured for the use of BGP Cost Community. Look at the diagram below, which shows PE's routing tables for VRF VPN_A.



Now assume that Loopback0 interface at CE1 goes down. The following is the step by step outline of the process that happens after this:

1. CE1 sends EIGRP queries to PE1 and CE2 asking for the prefix 172.16.1.1/32. Since PE1 prefers the prefix via CE1, it assumes there are no alternate path (queries are not propagated into BGP) and response with prefix unreachable to CE1.
2. At the same time, PE1's BGP table looks like this: the 172.16.1.1/32 prefix is locally sourced and the removal of EIGRP route will cause removal of BGP prefix as well:

```
PE1#show bgp vpnv4 unicast rd 100:4
BGP table version is 23, local router ID is 10.0.4.4
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
Route Distinguisher: 100:4 (default for vrf VPN_A)
*> 172.16.1.1/32    172.16.14.1        2688000         32768 ?
```


3. When the prefix is removed from PE1's BGP table, a BGP Withdraw message is sent to PE2 to signal the loss of prefix. Meanwhile, the query that CE1 sent to CE2 reaches PE2. At this point of time, further sequence of events depends on the race condition: whether BGP Withdrawn messages reaches PE2 first or the EIGRP query does that.
4. **BGP before EIGRP:** If the BGP message reaches PE2 *before* the EIGRP query for the same prefix, PE2 will remove it from the BGP table and consequently stops redistributing it into EIGRP. When the original EIGRP query reaches PE2, it has no alternate paths learned from BGP and responds that prefix is non-reachable. Eventually CE1 will receive replies from all routers and remove the prefix from its routing table. This is the desired outcome of events.
5. **EIGRP before BGP:** If the EIGRP query reaches PE2 before the BGP withdraw, PE2 thinks is has alternate path via BGP. It responds to CE2 with a new path and the reply travels back to CE1.

    a. CE1 now thinks it has path to 172.16.1.1/32 and sends UPDATE to CE2 and PE1. PE1 learns of the prefix again and advertises it into BGP.
    b. PE2 receives the BGP Withdrawn and realizes is lost all paths to 172.16.1.1/32. It sends out query to CE2 looking for alternate path. CE2 realizes it lost the only path to 172.16.1.1/32 and sends query to CE1. Lastly, CE1 realizes it lost the path again, and sends query to PE1.
    c. However, PE1 has already advertised the newly obtained information to PE2. By the moment the second query reaches PE1, it has already sent the update to PE2. Upon learning the loss of the prefix via EIGRP, PE1 sends Withdraw message to PE2 again.
    d. But now PE2 has learned path to 172.16.1.1 once more and sends an update message to CE2.

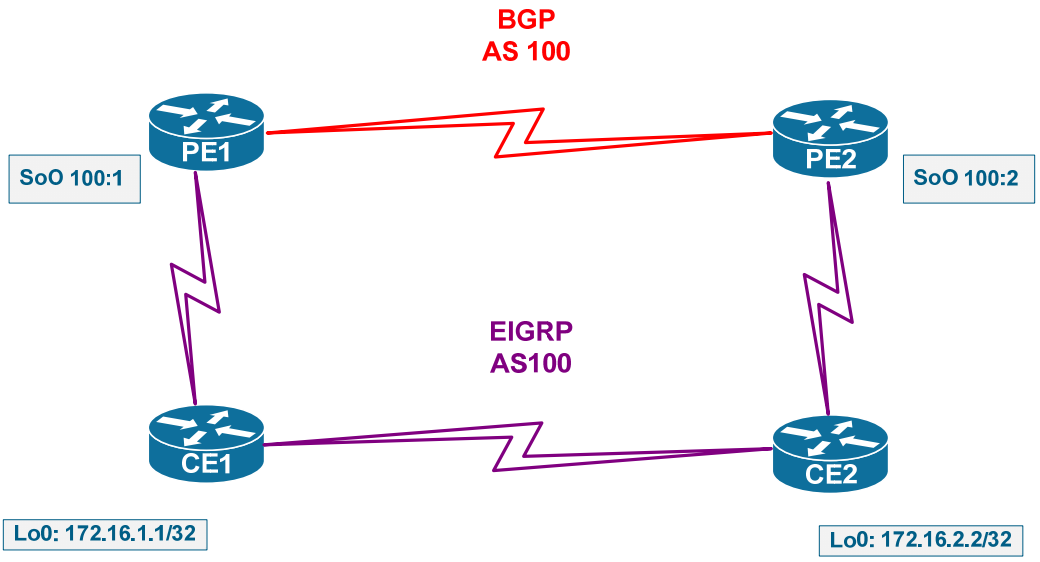
The cycle describe above may go infinitely. Due to the race condition between BGP and EIGRP message, stale information may circulate in the loop forever. There could be some variations to the above-outlined procedure, but all they are based on the race condition between BGP and EIGRP. Natively, EIGRP has a built-in mechanism to prevent this – maximum hop count (defined using the command **`metric maximum-hop`**). Since BGP preserves the original metrics, it also preserves the EIGRP hop count. Every time EIGRP router receives an update message it increments the hop-count by one. Eventually, the hop count will reach the maximum and the update will be dropped. This will break the loop, but will take some time for the topology to converge and age out stale information. The exact timing depends on the amount of time it takes for the BGP update to propagate from PE1 to PE2. This value is mainly affected by BGP process load, queueing delays, etc but mainly by the value of **`advertisement-interval`** set per peer.

There could be multiple solutions to this problem.

1. EIGRP maximum hops could be tuned to the lowest value possible. The problem is that picking this value could be troublesome, not to mention the other issue. The second drawback is that routing loop will persist for the amount of time it takes the stale information to circulate and age out. This could be in order or tens of seconds if BGP converges slowly.
2. The best solution is using tag-based route filtering. This is very similar to the use of route tagged for redistribution loop prevention, just based and some automatic IOS feature. This solution is known as EIGRP SoO (Site of Origin) community.

Deploying EIGRP SoO

There are two approaches to deploying this solution. We are going to illustrate them both and show how they work. The first approach is known as tagging EIGRP prefixes at PE routers only.



Here is how it works. The EIGRP SoO community applies on per-interface basis, using the sitemap construct. All EIGRP prefixes learned via this interface will carry the SoO extended community value. The EIGRP prefixes redistributed into BGP keep their SoO value in the BGP extended community. Conversely, all BGP routes redistributed into EIGRP and sent over the interface will have the SoO applied. There are two main rules for processing the SoO communities:

1. If the route is sent or received on the interface has the same SoO value as configured on the interface the route is discarded.
2. If the route sent or receive on the interface has an SoO value but it does not match the one configured for the interface, the value is preserved.

Now imagine that network has converged. At this moment, PE2 knows of CE1's prefix 172.16.1.1/32 via BGP and tags it with the SoO value of 100:1 (attached when the route was learned by PE1). Next, CE1's Loopback0 goes down, and the following occurs (assuming EIGRP wins the race condition with BGP):

1. R1's query reaches PE1 and PE1 responds with BGP information. This time, however, the route is tagged with SoO 100:1
2. The stale path reaches CE2 and CE1 and they both install it for a moment. Soon after this, PE2 loses the BGP path and sends out the EIGRP query.
3. Meanwhile, stale information reaches PE1. However, it is not redistributed into BGP, due to SoO filtering. The feedback loop stops, and PE2's query reaches CE1 and CE2 making them purge their stale paths.

In this scenario, stale information may exist in the topology for the amount of time it takes PE1 to update PE2 of the prefix loss via BGP. Not entirely perfect, but still better than relying on hop count. Now here is how you apply the SoO communities:

PE1:

```
route-map SITEMAP permit 10
 set extcommunity soo 100:1
!
interface Serial0/0/0.14 point-to-point
 ip vrf sitemap SITEMAP
```

PE2:

```
route-map SITEMAP permit 10
 set extcommunity soo 100:2
!
interface Serial0/0/0.25 point-to-point
 ip vrf sitemap SITEMAP
```


Here is how the SoO community attributes look like in the show command output:

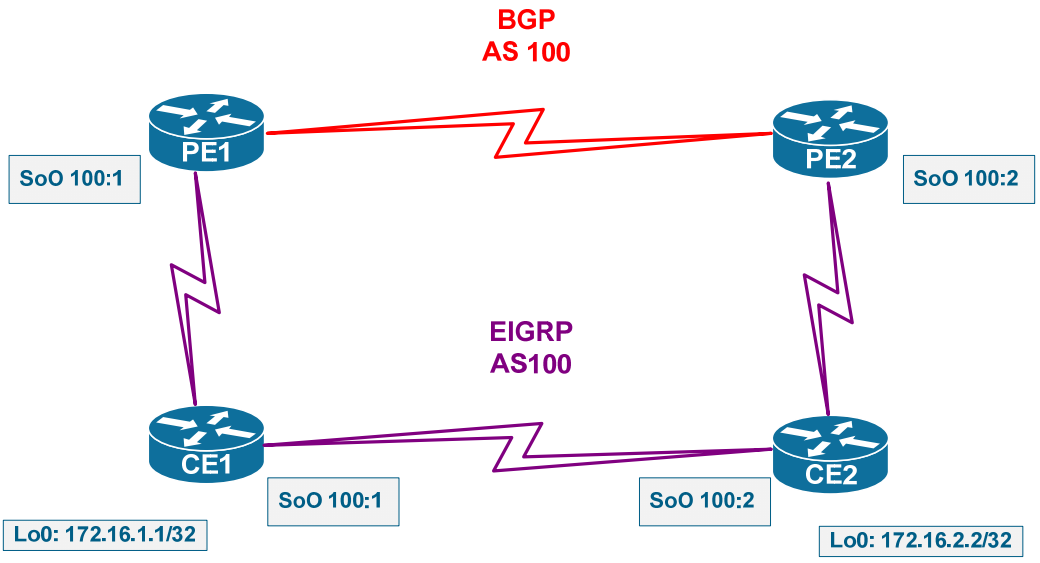
```
CE1#show ip eigrp topology 172.16.2.2/32
IP-EIGRP (AS 100): Topology entry for 172.16.2.2/32
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 128256
  Routing Descriptor Blocks:
  172.16.12.2 (Serial0/0.12), from 172.16.12.2, Send flag is 0x0
      Composite metric is (128256/128000), Route is Internal
      Vector metric:
        Minimum bandwidth is 1544 Kbit
        Total delay is 5010 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
  172.16.14.4 (Serial0/0.14), from 172.16.14.4, Send flag is 0x0
      Composite metric is (5248000/2688000), Route is Internal
      Vector metric:
        Minimum bandwidth is 1544 Kbit
        Total delay is 205000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
      Extended Community: SoO:100:2
```

CE1 sees CE2 Loopback0 prefix with the SoO value of 100:2 applied at PE2 while the following output:

```
CE2#show ip eigrp topology 172.16.1.1/32
IP-EIGRP (AS 100): Topology entry for 172.16.1.1/32
  State is Passive, Query origin flag is 1, 1 Successor(s), FD is 128256
  Routing Descriptor Blocks:
  172.16.12.1 (Serial0/0.12), from 172.16.12.1, Send flag is 0x0
      Composite metric is (128256/128000), Route is Internal
      Vector metric:
        Minimum bandwidth is 1544 Kbit
        Total delay is 5010 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 1
  172.16.25.5 (Serial0/0.25), from 172.16.25.5, Send flag is 0x0
      Composite metric is (5248000/2688000), Route is Internal
      Vector metric:
        Minimum bandwidth is 1544 Kbit
        Total delay is 205000 microseconds
        Reliability is 255/255
        Load is 1/255
        Minimum MTU is 1500
        Hop count is 2
      Extended Community: SoO:100:1
```

demonstrates that CE2 sees CE1 with the SoO value of 100:1. This ensures the validity of the configuration.

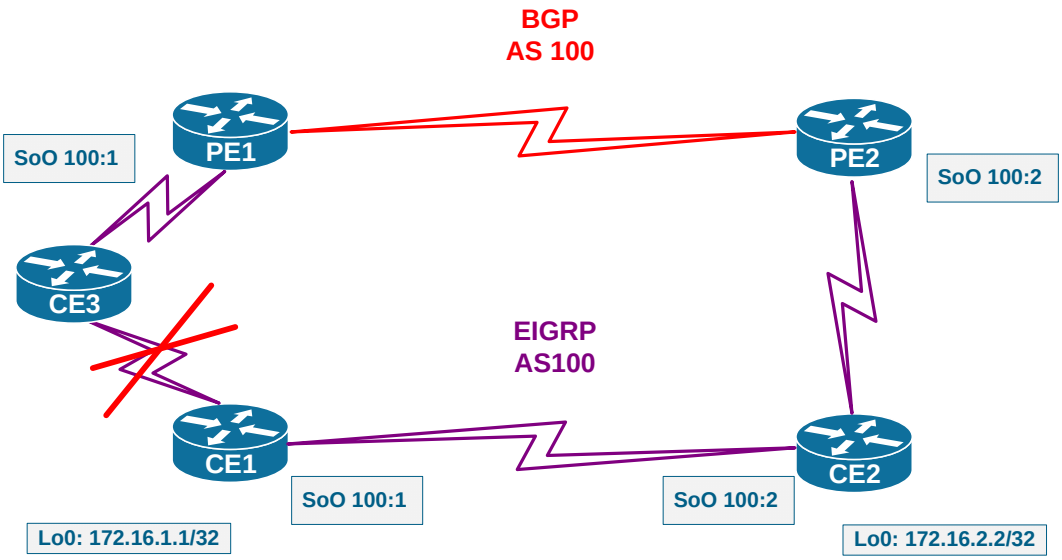
The second deployment scenario allows for almost completely eliminating transient routing loops trading off some redundancy. Here is the diagram:



In this case, SoO communities are applied on the PE-CE links as well as on the backdoor link. When a route flaps inside the VPN site, stale information is stopped from ever looping back to the origin as following:

1. CE1's Loopback0 interface goes down, and PE2 leaks stale information inside the VPN site. This stale prefix is tagged with SoO value of 100:1 as it was originated into BGP from R1.
2. The stale prefix is stopped at CE1, due to SoO filtering. CE1 never sees the prefix reviving and the routing loop never forms. For some time, CE2 may use the stale information until PE2 withdraws it

In this scenario, the only drawback is that the failure of the CE-CE link on one side of the topology may isolate the router connecting via the backdoor link from the isolate part connected to one of the PEs. On the diagram below, the failure of the link CE1-CE3 isolates CE1 from CE3 completely, as these routes cannot be learned across the backdoor link.



To summarize the above, you should always configure SoO community tagging for multihomed EIGRP connections, but it's up to you which scenario to choose. The second one offers more stable alternative, but less redundancy, while the first one may experience short periods of instability but provide complete redundancy. Lastly, it is interesting to note that the SoO community configuration is actually protocol agnostic, and you may see, for example, RIP routes getting SoO tagged as well, when transported inside MP-BGP.

Conclusions

Distance-vector protocols that utilize the gradient selection principle often exhibit various race conditions, which may lead to temporary routing loops. You may recall similar behavior found in RSTP and causing the well-know counting to infinity behavior. Since EIGRP is a distance-vector protocol, when coupled with BGP it exhibits the same problem in looped topologies. The use of IOS features such as SoO is almost mandatory if you want a stable network. The use of the Cost community is automatic and does not require and special configurations in Cisco IOS.

To verify your understanding of the issues discussed in this post, try answering the following questions:

1. Does OSPF need the Cost community to eliminate the same problem that EIGRP has?
2. Is there a way to make RIP prefer the SP path over a backdoor link?
3. Does RIP exhibit the same count-to-infinity problem that EIGRP has in multihomed scenarios?