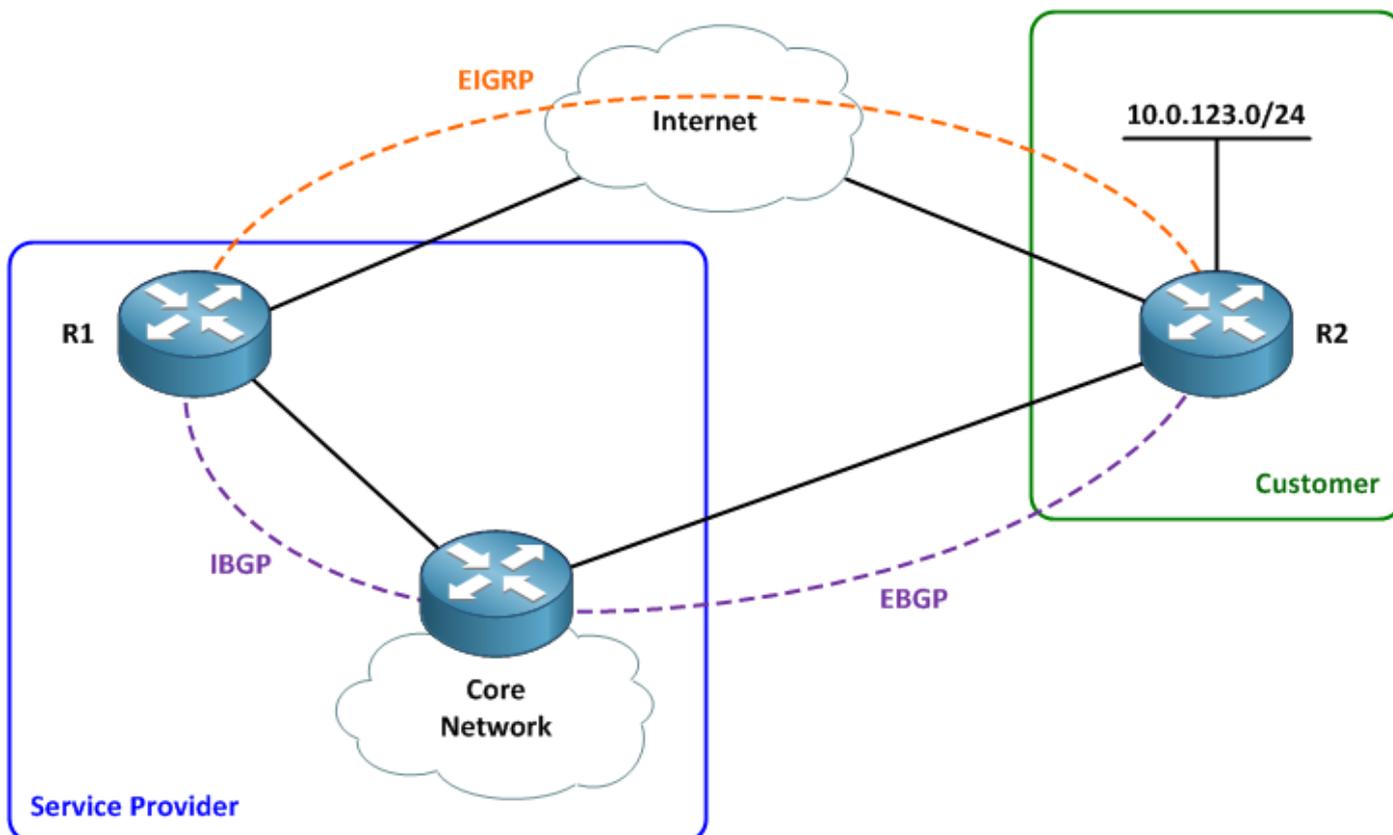


[A Slight Problem with Distributing EIGRP into MP-BGP](#)

By [stretch](#) | Wednesday, November 14, 2012 at 3:30 a.m. UTC

I ran into a problem recently when rolling out EIGRP on top of an Internet VPN for backup connectivity to customer sites, and the problem isn't even with the more complex bits of such an implementation. Cisco, it seems, has taken an unfortunately forward approach in attempting to design my network for me, stemming from a relatively obscure MP-BGP "feature."

Topology



Our lab topology features a service provider network and a customer site with a single router. The customer site has two paths to the provider: a dedicated circuit serves as the primary path and a backup [DMVPN tunnel](#) over the Internet provides failover connectivity. BGP is run across the dedicated circuit and EIGRP is run across the DMVPN tunnel.

The customer network exists as a VRF named ABC within the service provider network. The customer router advertises a single network (10.0.123.0/24) to the provider.

Initial Configuration

The basic configuration to bring up the router adjacencies might seem a bit tedious for those not accustomed to such a setup, but it's really pretty simple. First, we'll configure MP-BGP for the ABC VRF on R1 so that it can learn the 10.0.123.0/24 network advertised via the dedicated link toward the service provider network core.

R1

```
router bgp 100
no synchronization
bgp log-neighbor-changes
neighbor 172.16.0.5 remote-as 100
neighbor 172.16.0.5 soft-reconfiguration inbound
no auto-summary
!
```

```

address-family vpnv4
  neighbor 172.16.0.5 activate
  neighbor 172.16.0.5 send-community extended
exit-address-family
!
address-family ipv4 vrf ABC
  no synchronization
exit-address-family

```

We can see that BGP is up and the customer route has been learned via the core:

```

R1# show ip bgp vpnv4 vrf ABC 10.0.123.0
BGP routing table entry for 100:123:10.0.123.0/24, version 31
Paths: (1 available, best #1, table ABC)
Not advertised to any peer
65000
  172.16.0.5 from 172.16.0.5 (172.16.0.9)
    Origin IGP, metric 0, localpref 100, valid, internal, best
    Extended Community: RT:100:123
    mpls labels in/out nolabel/16

```

So far, so good. Next we'll bring up EIGRP. (We can disregard the VPN configuration since we're focused on routing for the purposes of this discussion.) Note that EIGRP is VRF-aware on R1.

R1

```

router eigrp 100
auto-summary
!
address-family ipv4 vrf ABC
  network 172.16.0.0 0.0.0.3
  auto-summary
  autonomous-system 65000
exit-address-family

```

R2

```

router eigrp 65000
network 10.0.123.0 0.0.0.255
network 172.16.0.0 0.0.0.3
no auto-summary

```

Finally, we'll redistribute EIGRP into MP-BGP on R1. We'll use a simple route-map to set the route's local preference to 90 (which is less than the default of 100) so that the BGP-learned route is preferred over the EIGRP-learned route.

```

route-map EIGRP-In permit 10
set local-preference 90
!
router bgp 100
address-family ipv4 vrf ABC
  redistribute eigrp 65000 route-map EIGRP-In
  no synchronization
exit-address-family

```

With our configuration work completed everything should function as desired. Unfortunately, this is where we run into a rather sticky problem. When we go to verify that the correct route is being taken, we see that it has actually disappeared: Only the EIGRP-learned route is present in the BGP table for VRF ABC.

```
R1# show ip bgp vpnv4 vrf ABC 10.0.123.0
BGP routing table entry for 100:123:10.0.123.0/24, version 36
Paths: (1 available, best #1, table ABC)
  Advertised to update-groups:
    1
Local
  172.16.0.2 from 0.0.0.0 (172.16.0.6)
    Origin incomplete, metric 409600, localpref 90, weight 32768, valid, sourced, best
    Extended Community: RT:100:123 Cost:pre-bestpath:128:409600
      0x8800:32768:0 0x8801:65000:153600 0x8802:65281:256000
      0x8803:65281:1500
    mpls labels in/out 18/nolabel
```

What gives? Let's look on one of the core routers.

```
Core# show ip bgp vpnv4 vrf ABC 10.0.123.0
BGP routing table entry for 100:123:10.0.123.0/24, version 30
Paths: (3 available, best #1, table ABC)
Flag: 0x820
  Advertised to update-groups:
    1
Local
  172.16.0.6 from 172.16.0.6 (172.16.0.6)
    Origin incomplete, metric 409600, localpref 90, valid, internal, best
    Extended Community: RT:100:123 Cost:pre-bestpath:128:409600
      0x8800:32768:0 0x8801:65000:153600 0x8802:65281:256000
      0x8803:65281:1500
    mpls labels in/out nolabel/18
65000
  172.16.0.10 from 172.16.0.10 (10.0.123.1)
    Origin IGP, metric 0, localpref 100, valid, external
    Extended Community: RT:100:123
```

Okay, there are our two routes. The path through the dedicated circuit (toward 172.16.0.10) has a local preference of 100, and the path through the VPN tunnel (toward R1 at 172.16.0.6) has a local preference of only 90, so the former route should be the best route, right? But here we see that the VPN route is clearly preferred. What's going on?

The Problem

The answer lies in this innocent-looking little extended BGP community:

```
Local
  172.16.0.6 from 172.16.0.6 (172.16.0.6)
    Origin incomplete, metric 409600, localpref 90, valid, internal, best
    Extended Community: RT:100:123 Cost:pre-bestpath:128:409600
      0x8800:32768:0 0x8801:65000:153600 0x8802:65281:256000
      0x8803:65281:1500
```

This is a type of BGP cost community, which are used to administratively influence the BGP path selection process. And this particular community is of the especially volatile *pre-bestpath* variety, which essentially hijacks the entire BGP path selection process. The community itself holds the metric carried over from the original EIGRP route, and because it is a pre-bestpath

community, this metric is evaluated before the traditional legacy BGP path selection algorithm starts.

Since the contending BGP route doesn't have a cost community assigned, it is considered to have a pre-bestpath metric of 2,147,483,647. This is the midpoint between 0 and 2^{32} , and obviously much, much higher than any sane EIGRP metric. So, the VPN route is declared the winner before BGP even gets a chance to weigh in (no pun intended).

But where did this cost community even come from? As it turns out, this behavior is actually baked into IOS to support a specific design scenario which Cisco calls [EIGRP MPLS VPN PE-CE with Backdoor Links](#):

The "pre-bestpath" point of insertion (POI) was introduced in the BGP Cost Community feature to support mixed EIGRP VPN network topologies that contain VPN and back door links. This POI is applied automatically to EIGRP routes that are redistributed into BGP. The "pre-best path" POI carries the EIGRP route type and metric. This POI influences the best path calculation process by influencing BGP to consider this POI before any other comparison step. No configuration is required. This feature is enabled automatically for EIGRP VPN sites when Cisco IOS Release 12.0(27)S is installed to a PE, CE, or back door router.

"No configuration is required." Well thanks, Cisco. That sure is a nifty trick you've got there, but how about you let *me* design my network? I'll tell the router if I want it to do stuff like that.

At this point you might naively hope, as I did, for a simple off switch for this "feature" given that this hot-wiring destroys the BGP path selection process we all know and love (or at least know). But there isn't one. I played around with route-maps and community strings for a few hours before I opened a case with Cisco TAC, who confirmed that, no, there isn't a way to disable this behavior.

Defeated, I had a few options to resolve the issue:

- Enable the command [bgp bestpath cost-community ignore](#) under the advisement of TAC on *all* core BGP peers. (Obviously this is not a trivial change.) This won't actually remove the community, but will neutralize its effect.
- Tweak the original EIGRP metric to something insanely high so that the default pre-bestpath cost is actually preferred.
- Drop EIGRP in favor of OSPF as my DMVPN routing protocol.

I opted for interim solution number two as it was admittedly dirty but by far the least disruptive, all things considered. Eventually, I'll implement `bgp bestpath cost-community ignore` across the core routers, or possibly even switch everything to OSPF just to avoid this mess altogether, but why should I have to? Am I unreasonable in expecting this to not be a default or behavior, or even if it is, that the IOS developers would include a mechanism to disable it?

I've submitted a [bug-report](#) feature request via TAC to implement some control over this behavior in the future, but I'm not holding my breath.

Posted in [Routing](#)