NOCTION
NETWORK INTELLIGENCE

BGP Prefix Filtering

Debugging BGP Prefix Propagation

www.noction.com

Table of Contents

Introduction

By default, BGP will simply propagate all the prefixes it knows to all neighboring routers. This is rarely what we want—without filters, a customer connected to two ISPs will start carrying traffic between those ISPs. Obviously the customer wouldn't want to use up their link capacity for third party traffic, and the ISPs also don't want this because the customer has no business looking at this traffic, and almost certainly lacks the bandwidth to handle it properly anyway. So customers only want to announce their own prefixes to Internet Service Providers (ISPs), and ISPs only want to accept a customer's prefixes from that customer. Although it would be preferable for ISPs to make sure that when they peer with other networks, they only accept the prefixes announced by that network and its customers, in practice it's hard to maintain such filters. So in peering relationships (interconnection without money changing hands), it's important that each peer only announces to peers the appropriate prefixes, as mistakes at this stage probably won't be caught further down the line. The appropriate prefixes to announce to a peer are a network's own prefixes and customer prefixes—but not prefixes from other peers or other ISPs.

The filtering set up on both ends of a BGP session reflects the business relationship between the two autonomous systems involved as shown in table 1.

	AS B announces only its own and its customer's prefixes to AS A	AS B announces all prefixes to AS A
AS A announces only its own and its customer's prefixes to AS B	ASes A and B are peers	AS B is provider, AS A is customer
AS A announces all prefixes to AS B	AS A is provider, AS B is customer	ASes A and B provide backup to each other (this is rare)

Table 1. Filtering and business relationships between autonomous systems

Prefix lists and AS path access lists from customers to ISPs

In an ideal world, there would be filters on all BGP sessions that only allow prefixes and AS paths that are actually supposed to come in over that BGP session. In practice, larger ISPs tend to get new IP address blocks and new customers with their own IP addresses at regular intervals, which makes maintaining such filters an uphill struggle. So more often than not, there's not much filtering on the peering sessions between ISPs. This means that once an ISP accepts a prefix advertised by a customer, that prefix will be widely propagated.

So it's absolutely critical that ISPs carefully filter prefixes they receive from their customers. A simple prefix list that allows the prefixes held by the customer (and the customer's customers) will do the trick:

```
!
ip prefix-list customer-a seq 5 permit 10.0.0.0/8 le 32
ip prefix-list customer-a seq 10 permit 172.16.0.0/12 le 32
ip prefix-list customer-a seq 15 permit 192.168.0.0/16 le 32
!
router bgp 65000
neighbor 192.0.2.2 remote-as 65001
neighbor 192.0.2.2 prefix-list customer-a in
!
```

The "le 32" part makes it possible for the customer to deaggregate their prefixes into smaller (more specific) prefixes. Doing that all the way down to /32s or individual IP addresses is not really useful, as typically, IPv4 prefixes longer than /24 aren't accepted. But if customers can label a prefix with an anti-DDoS community so the traffic towards that prefix is immediately dropped rather than delivered to the customer and clogging up their connection, it can be helpful to apply this on an individual /32 level. An outgoing filter that we'll discuss in a moment will make sure those /25s - /32s won't leave your network.

In addition to filtering prefixes, it's a good idea to also filter AS paths. That way, if one filter doesn't work, the other will still make sure only the right prefixes are propagated. Here, a customer with AS 65001 has two customers of their own: ASes 65002 and 65003:

```
!
ip as-path access-list 3 permit ^(65001_)+$
ip as-path access-list 3 permit ^(65001_)+(65002_)+$
ip as-path access-list 3 permit ^(65001_)+(65003_)+$
!
router bgp 65000
neighbor 192.0.2.2 remote-as 65001
neighbor 192.0.2.2 filter-list 3 in
!
```

AS path access lists use simple regular expressions to permit and deny AS paths. (65001_)+ means that the part between parentheses, 65001_, must occur one or more times. The underscore character can be the beginning of a line, the end of a line or the space between two AS numbers. So the lines "65001", "65001 65001", "65001 65001 65001" et cetera are all allowed by the first line of the AS path access list. Allowing each AS multiple times is important to make it possible for the customer to perform AS path prepending for traffic engineering. The next line allows one or more instances of the customer's AS and then one or more times the AS of the customer's first customer. The last line does the same for the customer's second customer.

Customers of ISPs (including ISPs that themselves buy transit service from a larger ISP) should use these same filters in the outgoing direction on their BGP sessions, with the caveat that the local AS number isn't added to the AS path until after AS path access lists have been applied. So without prepending, a locally originated prefix has an empty AS path. For AS 65001, the regular expression ^(65001_)*$ will allow those, with a * (zero or more times) rather than + (one or more times) applied to the (65001_) string.

Inbound prefix lists for everything else

As mentioned, between ISPs there's usually no strict filtering. Also, as customers expect to receive all prefixes reachable over the internet from their ISP(s), why have incoming filters on those BGP sessions?

However, even if you don't have a list of prefixes that you do want to receive from a peer or an ISP, there's still a list of prefixes that you don't want to receive from a peer or ISP: your own prefixes and internet exchange prefixes.

Suppose your prefix is 198.51.100.0/24 and you run your mail server on 198.51.100.13. So you announce 198.51.100.0/24. But what if a peer or ISP announces 198.51.100.13/32 to you? In that case, your routers will happily send all your mail towards your peer or ISP. So what you'll want to do is make a filter that doesn't allow your own prefixes or any more specific / longer prefixes (smaller address blocks) that fall within those prefixes:

```
!
ip prefix-list infilter seq 5 deny 198.51.100.0/24 le 32
ip prefix-list infilter seq 10 permit 0.0.0.0/0 le 24
!
router bgp 65000
neighbor 10.0.0.1 remote-as 65003
neighbor 10.0.0.1 prefix-list infilter in
!
```

This filter rejects 198.51.100.0/24 and all the prefixes that fall within 198.51.100.0/24 all the way to individual IP addresses. The second line then allows everything, as long as the prefix length is no more than /24. This is a filter that everyone should apply as an inbound filter on BGP sessions that don't have a stricter filter.

If your network is connected to one or more internet exchanges, you'll also want to reject the prefixes used for those internet exchange peering LANs in your inbound filters. The reason for this is that if someone announces a more specific prefix that falls within the internet exchange prefix, two bad things can happen. Your routers may send the traffic that should go towards the peers with addresses within that more specific prefix towards the network that announces the more specific rather than to the peers in question, because the next hop address learned from those peers is now routed to the source of the more specific prefix. Your router or routers that connect to the internet exchange in question will also try to send their BGP packets to the source of the more specific prefix rather than to the actual BGP neighbors with addresses in the more specific prefix. So after some time, those BGP sessions will go down. So reject all the peering LAN prefixes for internet exchanges that your network connects to in your inbound prefix list.

Resetting sessions

After building the right filters and applying them to the appropriate BGP sessions, there's one final step: resetting the BGP sessions. Changing filters on an active BGP session is like putting a bouncer in front of the door of a club when the club has been open for a while. Sure, no new people with the wrong shoes can get in, but the incorrectly dressed people who arrived before the bouncer was in place are still inside. So what's needed is the equivalent of getting everyone out and come back in again so the bouncer has the opportunity to see if they're dressed well enough. With BGP filters, we do this by clearing/resetting the BGP session.

The most straightforward way to reset a BGP session is with the ***clear ip bgp*** command.

NOTE: We'll use the old ***clear ip bgp*** ... and ***show ip bgp*** ... commands here, use ***clear bgp ipv6 unicast*** ... and ***show bgp ipv6 unicast*** ... for IPv6 BGP sessions and also ***clear bgp ipv4 unicast*** .... and ***clear bgp ipv4 unicast*** ... for IPv4 if you prefer. The arguments are the same for all three versions of the command.

All the ***clear*** commands take an IP address, an AS number or an asterisk as their first argument. Unsurprisingly, ***clear ip bgp 192.0.2.2*** clears (resets) the BGP session towards neighbor 192.0.2.2. ***clear ip bgp 65001*** clears all BGP sessions from this router towards routers in AS 65001. And ***clear ip bgp*** * clears all BGP sessions. Without further instructions, clearing a BGP session means that the session is torn down, all prefixes learned from the neighbor in question are removed from the BGP table and the routing table. The router then initiates a new BGP session. When that session is established, all prefixes are transferred and the new filter is applied.

WARNING: The downside of this procedure is that connectivity to and from the neighboring network is lost for some time; typically between 10 and 30 seconds, but longer is also possible. So clearing BGP sessions in this manner is ***not*** recommended. Especially as doing this counts as a "flap" and may land you on the receiving end of the flap damping mechanism, which can lead to periods of unreachability as long as half an hour. More about flap damping later.

Soft Resets

Fortunately, there's a better way. When you've changed an outgoing filter or route map, you can use the ***clear ip bgp ... out*** command. This doesn't reset the session, but rather, tells the router to push all the prefixes in the BGP table out to the neighbor or neighbors in question, applying the currently configured filters in the process. So prefixes that were previously filtered but are now allowed are then propagated to the neighbor, and prefixes that were allowed before but are filtered now are withdrawn. The result is exactly the same as with an actual reset of the BGP session, but without any disruption. The neighbor simply sees a sequence of updates. The ***clear ... out*** command doesn't require any cooperation from the BGP neighbor, so it always works, regardless of the capabilities of the neighbor.

It's also possible to push prefixes through the incoming filters and route maps using the ***clear ip bgp ... in*** command. The effect is the same as with the ***clear ... out*** command. However, the implementation is different: in order for the router to apply its current inbound filters and route maps, it has to receive new copies of all the neighbor's prefixes. It does this by asking the neighbor to send over its prefixes. This "route refresh" capability (***RFC 2918***) is an extension to the BGP protocol, so some routers don't implement it. You can determine this with the show ***ip bgp neighbors <address>*** command. The output may include the following:

```
Neighbor capabilities:
  4 Byte AS: advertised and received
  Route refresh: advertised and received(old & new)
```

The 4-byte AS capability refers to the ability to handle 32-bit AS numbers. As 32-bit AS numbers are in wide use today, all current routers support this capability. In this example, the local router advertises the route refresh capability and it also received an advertisement of the route refresh capability from the neighbor. "Old" refers to the old Cisco implementation of the capability before RFC 2918 was published, "new" means the RFC 2918 compliant implementation of route refresh.

Should your neighbor not support route refresh, all is not lost. It's possible to simulate this capability by configuring ***soft-reconfiguration inbound***:

```
!
router bgp 65000
 neighbor 192.0.2.2 remote-as 65001
 neighbor 192.0.2.2 soft-reconfiguration inbound
!
```

With this in place, the router will store a copy of all prefixes received from neighbor 192.0.2.2—including the ones that are filtered out. You can now use the ***clear ip bgp ... in*** or ***clear ip bgp ... soft*** in command to apply modified filters and route maps without a hard reset. Be warned that ***Cisco notes*** that "this method is memory-intensive and not recommended unless absolutely necessary."

Inspecting the BGP table

However, ***soft-reconfiguration inbound*** does have an added benefit: you can inspect the full set of prefixes advertised by a neighbor, including all the prefixes filtered out. This is done with the ***show ip bgp neighbors ... received-routes*** command. This lists all the prefixes learned from the neighbor in question in the same format as the ***show ip bgp*** output. This includes prefixes that are filtered out which aren't added to the BGP table. ***The show ip bgp ... routes*** command, on the other hand, only shows the prefixes learned from the neighbor that are actually added to the BGP table. Another difference is that ***show ... received-routes*** shows the prefixes with their original attribute values, while show ... routes shows them after a route map may have altered certain attributes. For instance, this route map changes most of a prefix's BGP attributes:

```
!
route-map makechanges permit 10
 set as-path prepend 4200000000
 set metric 12345
 set local-preference 10000
 set ip next-hop 192.0.2.222
 set weight 999
!
```

Which results in:

```
Router# show ip bgp neighbors 192.0.2.2 routes
   Network    Next Hop     Metric   LocPrf   Weight   Path
*> 2.0.0.0    192.0.2.222  12345    10000    999      4200000000 2 i
```

The ***show ... received-routes*** command, on the other hand, shows the prefix in its original form:

```
Router# show ip bgp neighbors 192.0.2.2 received-routes
   Network     Next Hop     Metric   LocPrf   Weight   Path
*> 2.0.0.0     192.0.2.2    0        10000    0        2 i
```

More detailed information on prefixes in the BGP table (but not ones filtered out and kept by ***soft-reconfiguration inboud***) is shown using the ***show ip bgp <prefix>*** command:

```
Router# show ip bgp 2.0.0.0
BGP routing table entry for 2.0.0.0/8
Paths: (2 available, best #2, table Default-IP-Routing-
Table)
  Not advertised to any peer
  4200000000 2
    192.0.2.222 from 192.0.2.2 (99.83.0.1)
     Origin IGP, metric 12345, localpref 10000, weight 999,
valid, external
  2
    203.0.113.5 from 203.0.113.5 (199.84.0.1)
      Origin IGP, localpref 10000, valid, external, best
      Community: 65000:2 65000:3
```

In this example, no prefix length was specified for network 2.0.0.0, but the router found prefix 2.0.0.0/8—it didn't automatically assume we meant /8 because 2.0.0.0 is a class A network. There are two paths/routes available towards this prefix. The first route is the one mangled by the route map above, as we can see from the presence of AS 4,200,000,000 in the AS path. On the next line, the first address is the next hop address. This is usually the same as the next address, the neighbor address. But in this case the route map changed the next hop address, but the neighbor address isn't affected. The third address is the router ID of the neighbor, which we can use to determine whether two BGP sessions go towards the same router or to different routers.

The second path towards 2.0.0.0/8 is considered best by BGP; it also has the very high local preference value of 10,000, but the AS path is only one hop rather than two. This path also has two community values attached.

The router indicates that the prefix is not announced to peers (neighbors). If it is, you'll usually see their IP addresses here. So the ***show ip bgp <prefix>*** command is useful to see whether a specific prefix is announced to neighbors. Alternatively, you can use the ***show ip bgp neighbors ... advertised-routes*** command to list all the prefixes announced to a specific neighbor.

Propagation of updates

BGP is a real-time system, so when a BGP session is established or a new prefix is injected into BGP, the new information typically propagates very quickly. You can monitor this using one of the many "looking glasses" that are available around the globe. (Look for "BGP looking glass" in a search engine.) However, sometimes it takes a little longer of prefixes to propagate to all corners of the internet. The main reason for this is the minimum route advertisement interval (MRAI). ***RFC 4271*** specifies that a certain minimum amount of time should elapse between eBGP updates for "routes to a particular destination". The suggested value for the MRAI is 30 seconds.

The idea is that if there's an update for the same prefix every five seconds, the first update is propagated, and then nothing happens for the next 30 seconds. After the 30 seconds, the most recent information is sent to the neighbor or neighbors in question. So the updates at 5, 10, 15, 20 and 25 seconds aren't propagated. This reduces the instability that may be injected into the BGP system.

When a new prefix is announced because it's newly injected into BGP or a session that has been down comes back up, usually the MRAI doesn't come into play, because the update typically travels fastest over the best path. So that best path is installed in the BGP table and used. The update then comes in over longer paths, but those updates aren't propagated because they contain paths that are worse than the already known path. (Exceptions are possible when for traffic engineering or policy reasons, a path that's longer than the shortest path is preferred.)

However, this is different when routes are removed from BGP, and to a lesser degree when routes are made less preferred, for instance, by path prepending. In that case, BGP will start "path hunting". Consider the topology in ***Figure 1***.

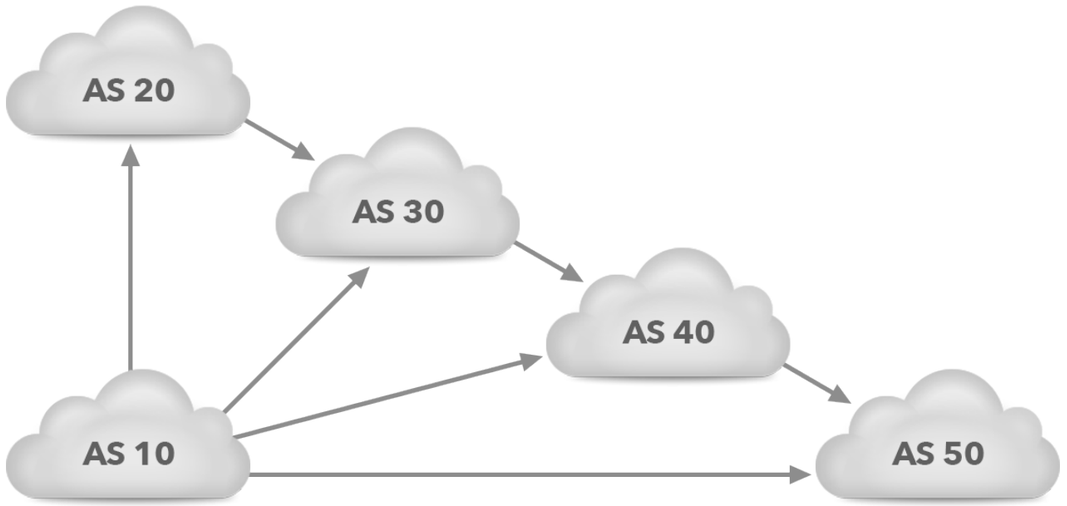

Figure 1. Path hunting example topology

ASes 20, 30, 40 and 50 all receive the prefix that AS 10 advertises. AS 20 propagates the prefix to AS 30, AS 30 propagates it to AS 40, and AS 40 propagates it to AS 50. This results in the BGP tables shown in ***Table 2***. The greater-than sign indicates the best path.

AS	PREFIX	AS PATH
20	198.51.100.0/24	> 10
30	198.51.100.0/24	20 10 > 10
40	198.51.100.0/24	30 10 > 10
50	198.51.100.0/24	40 10 > 10

Table 2. Initial AS paths

When at this point AS 10 goes down or otherwise stops advertising prefix 198.51.100.0/24, each AS will remove the direct path towards AS 10, as shown in ***Table 3***.

AS	PREFIX	AS PATH
20	198.51.100.0/24	
30	198.51.100.0/24	> 20 10
40	198.51.100.0/24	> 30 10
50	198.51.100.0/24	> 40 10

Table 3. AS paths immediately after AS 10 becomes unreachable

Now, the following updates will happen:

- **AS 40 tells AS 50 that the path is now 30 10**
- **AS 30 tells AS 40 that the path is now 20 10**
- **AS 20 tells AS 30 that 198.51.100.0/24 is now unreachable**

After this round of updates, the BGP tables look as shown in ***Table 4***.

AS	PREFIX	AS PATH
20	198.51.100.0/24	
30	198.51.100.0/24	
40	198.51.100.0/24	> 30 20 10
50	198.51.100.0/24	> 40 30 10

Table 4. AS paths after the second round of updates, shortly after AS 10 becomes unreachable

Immediately after receiving its update from AS 30, AS 40 needs to send another update to AS 50, informing it that the path is now even longer. And AS 30 no longer has a route towards 198.51.100.0/24, so it needs to send a withdrawal towards AS 40.

However... This is where the minimum route advertisement interval kicks in, as AS 30 already just sent an update to AS 40 and AS 40 one to AS 50. So for 30 seconds, nothing happens. Then the updates are sent, resulting in the BGP tables shown in ***Table 5***.

AS	PREFIX	AS PATH
20	198.51.100.0/24	
30	198.51.100.0/24	
40	198.51.100.0/24	
50	198.51.100.0/24	> 40 30 20 10

Table 5. AS paths after the third round of updates, 30 seconds after AS 10 becomes unreachable.

AS 40 now sees the withdrawal from AS 30 and needs to propagate that withdrawal to AS 50. But the MRAI delays the update once again, so this withdrawal is delayed by another 30 seconds. After that delay, we reach a new stable situation where 198.51.100.0/24 is considered unreachable by every AS, as shown in ***Table 6***.

AS	PREFIX	AS PATH
20	198.51.100.0/24	
30	198.51.100.0/24	
40	198.51.100.0/24	
50	198.51.100.0/24	

Table 6. AS paths after the fourth round of updates, 60 seconds after AS 10 becomes unreachable

NOTE: The MRAI is intended to work on individual prefixes, but implementations may group prefixes together and apply the MRAI to the group in order to save bookkeeping overhead. As a result, the MRAI may slow down a prefix even if it's updated only once rather than multiple times.

So the path hunting behavior means that when a prefix becomes unreachable, BGP will explore longer and longer paths before the prefix disappears from routing tables everywhere. When a prefix really becomes unreachable, it's not a problem that for about two minutes, packets try to follow longer and longer paths before they're ultimately lost anyway. However, path hunting can be very problematic when the prefix in question is a more specific prefix, and there's also a less specific prefix covering the same address range.

For instance, suppose that a network has prefix 198.51.100.0/23 and is connected to two ISPs. In order to get their traffic engineering to work the way they want, they decide to split the /23 into two /24s, and announce 198.51.100.0/24 to one ISP and 198.51.101.0/24 to the other ISP. The /23 is announced to both ISPs as a backup. If now the link to the first ISP goes down, path hunting will happen for 198.51.00.0/24. Once the path hunting is over, traffic will flow towards the /23, but during path hunting, there will be instability, as paths keep changing every 30 seconds. Some of these paths may work, others may not. Typically, this takes two minutes, with pings coming through during some of the 30-second periods and going unanswered during others.

NOTE: So it's best to not depend on a less specific advertisement as a backup for a more specific advertisement. In the situation with two ISPs, it would have been better if each /24 were advertised to both ISPs, but with different properties to influence traffic engineering. However, it's possible for prefixes announced with many prepends to still receive incoming traffic, so this option isn't always ideal, either.

Flap damping

In the late 1990s, there were issues with BGP implementations that caused large numbers of "flaps". A flap is a change to an existing prefix: a withdrawal or an update that modifies one or more attribute values. An update that installs a path through a neighbor where the neighbor previously didn't have a path is not considered a flap. And presumably, BGP implementations are smart enough to not send/propagate updates for an existing prefix when all attributes remain the same.

Back some 20 years ago, routers used relatively even more underpowered CPUs than today, and the persistent flapping of significant numbers of prefixes created high CPU loads in numerous routers. ***RFC 2439*** specifies flap damping, a mechanism to protect routers from excessive flapping.

Flap damping works by assigning a penalty to a prefix every time it flaps. When the penalty rises above a preset value, the prefix in question is "suppressed" and temporarily not propagated towards BGP neighbors. The penalty decreases though an exponential decay function. If flapping doesn't continue, the penalty eventually reaches the reuse-limit and the prefix is no longer suppressed and propagated over BGP again. When the penalty reaches half the reuse-limit, the damping information for the prefix is removed from memory.

Cisco's ***implementation of path damping*** uses fixed penalties of 1000 for withdrawals and 500 for attribute changes. By default, the suppress-limit is 2000 and will thus be triggered by three withdrawals or five attribute changes in a short time. The penalty decays with a half-life of 15 minutes; the reuse-limit is 750 and there's a max-suppress-time of 60 minutes by default. So three withdrawals within a few minutes will create a penalty of nearly 3000, which needs to halve twice to reach the reuse-limit, so the prefix will be damped for about half an hour.

When different networks use different flap damping settings—and the RFC doesn't specify default values—there is the possibility that the flap damping in one network triggers further flap damping in another network. This, combined with path hunting, may lead to unnecessary and possibly excessive periods of unreachability. To avoid this, in 2001, RIPE published a document with suggested flap damping parameters. If all networks use these coordinated parameters, which take the prefix length into account, they'll all damp in the same way and unintended side effects are avoided.

A few years later, the recommendation changed to disabling flap damping "because it created more harm than good". Currently, ***RIPE document 580*** recommends that network operators, if they want to use flap damping, simply set the suppress threshold to at least 6000.

BGP implementations are much less prone to generating continuous flaps these days, and despite the growth in the BGP table, modern router CPUs can keep up with BGP update rates. Also, as observed by RIPE, there is the potential for harmful effects. And a router performing flap damping actually has to spend more CPU cycles processing flaps in order to protect routers further upstream. So flap damping is no longer in wide use these days.

WARNING: However, some networks still perform flap damping. Resetting a BGP session a couple of times within a short timeframe can easily suppress the prefixes involved for half an hour or longer in a network that uses flap damping. As such, it's still prudent to avoid hard resets of BGP sessions and especially multiple hard resets within a short time. Changes that generate updates with modified attribute values are also best kept to a minimum.

NOCTION
NETWORK INTELLIGENCE

This ebook was brought to you by **Noction**.

Noction Intelligent Routing Platform enables enterprises and service providers to maximize end-to-end network performance and safely reduce infrastructure costs. The platform evaluates critical network performance metrics in real-time and responds quickly by automatically rerouting traffic through a better path to avoid outages and congestion.

Request a free trial today and see how IRP can boost your network performance.

Start a Free Trial