Improving BGP Convergence

Daniel Walton

1.0 - Introduction

As the Internet table has grown, service providers and large enterprise customers are noticing a dramatic increase in the amount of time BGP takes to converge. Networks that once converged in 10 or 15 minutes may now take up to one hour in some cases and may take even longer in extreme situations. Longer convergence times are due to the increased size of the Internet table and an increase in the number of peers supported by a single BGP speaker.

A full BGP feed now includes over 100k routes but only included 60k routes three or four years ago. Today a route-reflector may have more than 100 route-reflector-clients where three or four years ago it may have had 50. More routes and more peers means that BGP has a lot more work to do today than in the past. The end results are the extreme convergence times we are seeing in large networks.

The goal of this project is to discuss how to improve BGP convergence by config modifications and changes in Cisco's BGP implementation. We hope customers can use this paper to improve BGP convergence in their network.

2.0 - Test Description

In a nutshell, routes will be accepted from a single eBGP "Feeder" peer and advertised to X eBGP peers in a single peer-group. The goal is to see how large X can be and still converge within 10 minutes. The number of routes we are receiving from our "Feeder" peer will play an important roll in how many peers we can converge in 10 minutes so we will test with 80, 90k, 100k, 110, and 120k routes.

Note: A 7200VXR with NPE400 was used for all tests described in this paper.

3.0 - Definitions

Converged - BGP is considered converged when all of the following are true:

- All routes have been accepted
- All routes have been installed in the routing table
- The table version for all peers must be equal to the table version of the BGP table
- The InQ and OutQ for all peers must be zero

Convergence Time - The amount of time BGP takes to converge. The timer starts as soon as the first BGP peer is established and stops as soon as BGP is converged (see def. above).

READ_ONLY Mode - If BGP is in READ_ONLY mode then BGP is only accepting routing updates and is not computing a best path nor advertising routes for any prefixes. When the BGP process starts (i.e. after a router reboot) BGP will go into READ_ONLY mode for a maximum of two minutes. RO mode forces a BGP speaker to sit still for a few minutes giving his peers a chance to send their initial set of routes.
updates. The more routes/paths BGP has the more stable the network will be because we will avoid the scenario where BGP sends an update for a prefix and then learns about a better path for that prefix a few seconds later. If that happens then BGP sent two updates for a single prefix, which is very inefficient. READ_ONLY mode increases the chances of BGP learning about the bestpath for a prefix before sending out any advertisements for that prefix. BGP will transition from RO mode to RW mode once all of our peers have sent us their initial set of updates or the two-minute RO timer expires.

**READ_WRITE Mode** - This is the normal mode of operation for BGP. While in READ_WRITE mode BGP will install routes in the routing table and will advertise those routes to his peers.

**peer-group synchronization** - Refers to a peer-group member and the peer-group leader for a peer-group being synchronized in terms of the routes that have been advertised to both. Example: If BGP has only advertised routes XYZ to the peer-group leader then a peer-group member is said to be "in sync" if he has also only been advertised routes XYZ.

**UPDATE Packing** - A BGP UPDATE consists of a combination of attributes (MED = 50, LOCAL_PREF = 120, etc.) and a list of NLRI (Network Layer Reachability Information) that share that combination of attributes. The more NLRI BGP is able to list in an UPDATE the faster BGP will be able to converge because overhead (IP, TCP, BGP Headers, etc.) will be reduced. "UPDATE packing" refers to the packing of NLRI into BGP UPDATEs. Example: If a BGP table has 100k routes and 15k different attribute combinations then BGP will only need to send 15k UPDATEs if NLRI is packed with 100% efficiency. If BGP were packing with 0% efficiency then we would need to send 100k UPDATEs. Obviously it is better to advertise 100k routes via 15k UPDATEs than 100k UPDATEs. This can be checked by looking at the number of messages formatted as shown by "show ip bgp peer-group".

**UPDATE Replication** - If a peer-group member is "in sync" then we will take an UPDATE message that was formatted for the peer-group leader and replicate it for the peer-group member. It is far more efficient to replicate an update for a peer-group member than it is to re-format the update. Example: Assume we have 20 members in a peer-group and we need to send 100 messages. If we format 100 messages for the peer-group leader and replicate those messages to the other 19 peer-group members then we are achieving 100% replication. This can be checked by comparing the number of messages replicated to the number of messages formatted as shown by "show ip bgp peer-group".

**Input Queue drops** - Traffic that is destined to the router must come through an interface input queue ("show interface" will show this queue). Characteristics of the interface input queue are:

- By default this queue is 75 packets deep plus 100 spots in the SPD headroom
- BGP uses IP Prec 6 so it is possible to see the input queue at 176/75
- The only packets that are allowed in the SPD headroom are "important" packets with IP Prec of 6 or 7
- If the input queue gets to 176/75 and another important packet comes in then that packet will be dropped and the "input drop" counter will be incremented. This type of drop will also be counted as a "SPD Priority Drop" in "show interface switching".

4 - Convergence Improvements

4.1 - BGP/TCP Interaction
CSCdr50217 "BGP: Sending updates slow" addresses a bug in the BGP->TCP code that caused BGP to converge slowly. Prior to CSCdr50217 BGP would queue data from the BGP OutQ to the TCP socket for each peer once every second. After CSCdr50217, BGP will no longer wait one second but will aggressively queue data from the OutQs into the TCP sockets until the OutQs have been completely drained. This allows BGP to send data at a much faster rate, which allows BGP to converge faster. You can see in the following graph that the number of supported peers more than doubled in releases of IOS that have this bug fix:

![Graph showing supported peers vs number of routes]

4.2 - Peer Groups

Perhaps the most important thing that can be done to improve BGP scalability is to use peer-groups. Peer-groups are normally used as a means of simplifying BGP configuration but they provide scalability assistance as well. All members of a peer-group are required to share a common outbound policy, which means that the same UPDATEs can be delivered to each peer-group member. The ability to replicate the same UPDATE to multiple peers reduces the amount of time and the number of CPU cycles that BGP requires to advertise routes to peers:

- UPDATE generation without peer-groups - BGP walks the table for every peer, prefixes are filtered through outbound policies, UPDATEs are generated and sent to this one peer

- UPDATE generation with peer-groups - A peer-group leader is elected for each peer-group. BGP walks the table for the leader only, prefixes are filtered through outbound policies, UPDATEs are generated and sent to the peer-group leader and replicated for peer-group members that are synchronized with the leader

Comparing the number of supported peers with peer-groups vs without shows a 30% to 50% increase from 4.1. Recommendations we make later in this paper may show bigger jump in terms of percentages but peer-groups are still the #1 change that can be made to improve convergence. Some of the other recommendations such as "ip tcp path-mtu-discovery" address bottlenecks that exist because peer-
groups are used. Enabling mtu-discovery, upgrading code, and increasing input queues will not have the significant benefits that you see in this paper if peer-groups are not used! Note: unless otherwise noted, peer-groups will be used for the remaining tests in this paper.

4.3 - "ip tcp path-mtu-discovery"

Every TCP session has a limit in terms of how much data it can transport in a single packet. This limit is defined as the Maximum Segment Size (MSS) and is 536 bytes by default. This means TCP will take all of the data in a transmit queue and break it up into 536 byte chunks before passing packets down to the IP layer. Using a MSS of 536 bytes ensures that the packet will not be fragmented before it gets to its destination because most links have a MTU of at least 1500 bytes. The following command will allow you to check the MSS of all BGP peers:

```
Router#show ip bgp neighbors | include max data
Datagrams (max data segment is 536 bytes):
Datagrams (max data segment is 536 bytes):
Datagrams (max data segment is 536 bytes):
Datagrams (max data segment is 536 bytes):
```

The problem is that using such a small MSS value creates a large amount of TCP/IP overhead, especially when TCP has a lot of data to transport like it does with BGP. The solution is to dynamically determine how large the MSS value can be without creating packets that will need to be fragmented. This is accomplished by enabling "ip tcp path-mtu-discovery" (a.k.a. PMTU). PMTU allows TCP to determine the smallest MTU size among all links between the ends of a TCP session. TCP will then use this MTU value minus room for the IP and TCP headers, as the MSS for the session. If a TCP session only traverses Ethernet segments then the MSS will be 1460 bytes. If it only traverses POS segments then the MSS will be 4430 bytes. The increase in MSS from 536 to 1460 or 4430 bytes

```
12.0(18)S - no peer-groups  12.0(18)S - peer groups
```
reduces TCP/IP overhead, which helps BGP converge faster. Once we have enabled this knob and reset all of our BGP sessions via a reboot or "clear ip bgp *" we can see the expected increase in MSS:

```
Router#show ip bgp neighbors | include max data
Datagrams (max data segment is 1460 bytes):
Datagrams (max data segment is 1460 bytes):
Datagrams (max data segment is 1460 bytes):
Datagrams (max data segment is 1460 bytes):
```

The following graph shows a leap in the number of supported peers when enabling PMTU. Testing was done over GigE links so the MSS increased from 536 bytes to 1460 bytes.

![MTU Discovery Graph]

**4.4 - Increasing Interface Input Queues**

Large numbers of interface input queue drops are a very common problem for routers with more than fifteen or twenty BGP peers. When BGP is advertising thousands of routes to many peers TCP will transmit thousands of packets in a short period of time. Our BGP peers will receive these packets and send TCP acknowledgements to the advertising BGP speaker which will cause the BGP speaker to receive a flood of TCP acks in a short period of time. The acks will come in at a rate faster than the route-processor can handle which will cause us to buffer them in the interface input queue. This queue is only 75 spots deep and the SPD (Selective Packet Discard) headroom is 100 spots deep. TCP acks can quickly fill the 175 spots of input buffering, which will lead to massive amounts of dropped packets. It is common to see 10,000+ drops per interface per minute if BGP is advertising a full Internet table to many peers. Example from a router 15 minutes after reboot:

```
Router#show int pos 8/0 | include input queue
Output queue 0/40, 0 drops; input queue 0/75, 278637 drops
Router#
```

Increasing the interface input queue depth (`hold-queue <1-4096> in`) will help reduce the number of dropped TCP acknowledgements which reduces the amount of work BGP has to do to converge. The
following graph shows that we can support twice as many peers by increasing the interface input queue depth from 75 to 4096.

4.5 - "ip tcp path-mtu-discovery" and Larger Input Queues

Combining PMTU with larger input queues will make a huge difference in BGP convergence. Here we can see how these simple configuration changes allow BGP to support 3x as many peers.
4.6 - Software Improvements in 12.0(19)S

Even with larger input queues and PMTU a few customers still have BGP convergence problems. Some large ISPs carry two times as many routes (~250,000) as there are in the Internet table, which makes converging difficult. Advertising 250k routes to even a small number of peers is a challenging task but when you have to push these routes out to hundreds of peers BGP has a lot of work to do. BGP as we know it in 12.0(18)S could not effectively handle advertising such a large number of routes to so many peers. Several optimizations were made to the BGP peer-group code to improve UPDATE packing and UPDATE replication. The improvements make a dramatic difference in convergence times and allow BGP to support many more peers.

Example:
The efficiency of UPDATE Packing and UPDATE Replication can be checked by the "show ip bgp peer-group" command. The following output is from a convergence test with 6 peer-groups, 20 peers in each of the first five peer-groups (eBGP peers) and 100 peers in the sixth peer-group (iBGP peers). Also, the BGP table that was used had 36,250 attribute combinations.

"show ip bgp peer-group | include replicated" on 12.0(18)S gives us the following information:

Update messages formatted 836500, replicated 1668500  
Update messages formatted 1050000, replicated 1455000  
Update messages formatted 660500, replicated 1844500  
Update messages formatted 656000, replicated 1849000  
Update messages formatted 501250, replicated 2003750  
Update messages formatted 2476715, replicated 12114785

To calculate the replication rate for each peer-group divide the number of updates replicated by the
number of updates formatted:

1668500/836500 = 1.99  
1455000/1050000 = 1.38  
1844500/660500 = 2.79  
1849000/656000 = 2.81  
2003750/501250 = 3.99  
12114785/2476715 = 4.89

If BGP replicated perfectly then the eBGP peer-groups would each have a replication rate of 19 because there are 20 peers in the peer-group. The update should be formatted for the peer-group leader and then replicated to the other 19 peers giving us an optimal replication rate of 19. The ideal replication rate for the iBGP peer-group would be 99 since we have 100 peers.

If BGP packed updates perfectly then we would have only formatted 36250 updates. We should only need to generate 36250 updates for each peer-group because that is the number of attribute combinations in the BGP table. The iBGP peer-group alone formats almost 2.5 million updates while the eBGP peer-groups each generate anywhere from 500k to 1 million updates.

"show ip bgp peer-group| include replicated" on 12.0(19)S gives us the following information:

UPDATE packing is optimal, we are formatting exactly 36250 updates for each peer-group:

Update messages formatted 36250, replicated 688750
Update messages formatted 36250, replicated 688750
Update messages formatted 36250, replicated 688750
Update messages formatted 36250, replicated 688750
Update messages formatted 36250, replicated 688750
Update messages formatted 36250, replicated 688750
Update messages formatted 36250, replicated 688750
Update messages formatted 36250, replicated 3588750

UPDATE replication is also perfect:

688750/36250 = 19
688750/36250 = 19
688750/36250 = 19
688750/36250 = 19
688750/36250 = 19
3588750/36250 = 99
5.0 - Conclusion

There are several steps that customers can take to improve convergence without upgrading IOS. Using "ip tcp path-mtu-discovery" and larger input queues require only a few lines of config changes but will greatly reduce the amount of time BGP takes to converge. Using peer-groups requires a little more effort but will make large numbers of peers easier to manage due to a smaller config and will also provide many scalability benefits.

If customers are willing to upgrade, using 12.0(19)S will give them several BGP bug fixes that make BGP much more scalable. Using 12.0(19)S with peer-groups, PMTU, and larger input queues will allow BGP to scale well beyond 1000 peers!!
Please send questions or comments to rp-deployment@cisco.com
Last modified 12-12-2001