Mini Project
BGP Prefix Prioritization from pcap

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Junos ® BGP Prefix Prioritization – from pcap

1.0

Table of Contents

BGP Route Prioritization

We will study BGP route prioritization feature for this. The Border Gateway Protocol (BGP) is used not only as the Internet's inter-domain routing protocol, but also as the primary protocol to carry the many types of Virtual Private Network reachability, including Layer 2 VPN (VPLS), Layer 3 VPN, Multicast VPN, Ethernet VPN, etc. This increased pressure on BGP - especially in scaled Provider Edge, eBGP or Router Reflector roles - often means that a BGP speaking router has a significant amount of Network Layer Reachability Information (NLRI; i.e. routes) carried in the form of BGP Updates to exchange.

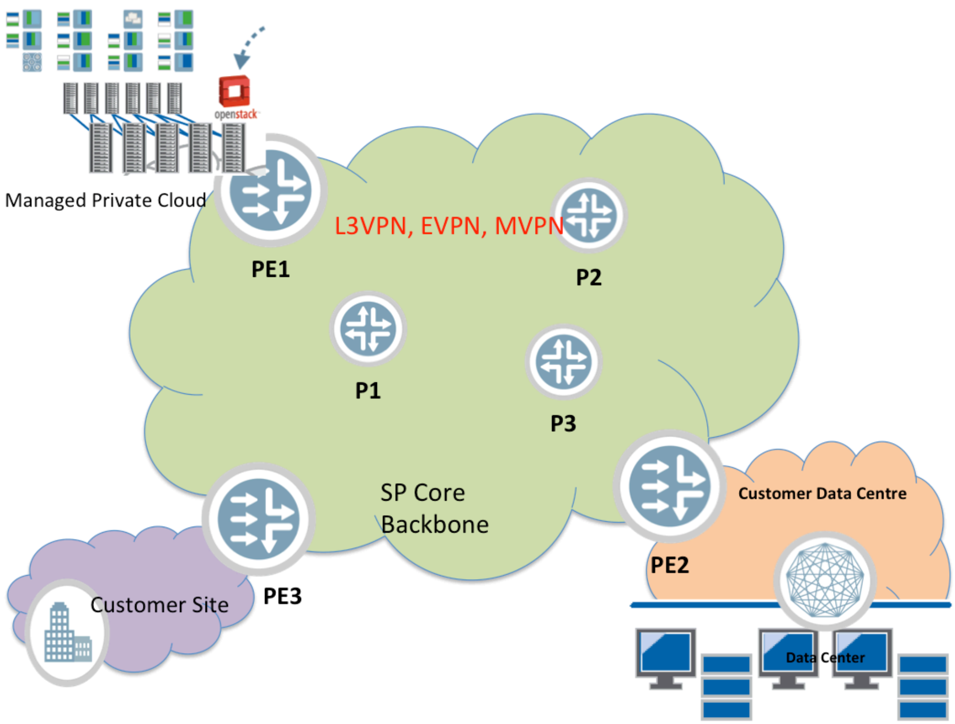


Fig 01: BGP Services on the SP core

BGP uses the TCP protocol to carry its routing information. TCP provides reliability and sequencing for the protocol, but is a stream protocol that requires BGP to provide the underlying framing.

While the BGP specifications provide guidance as to how routing information is encoded in BGP messages, it doesn't provide specific guidance as to how to order that information while exchanging routes in the form of BGP Update messages. Instead, it leaves this up to the implementation. Especially in scaled systems, BGP may take a significant amount of time to exchange its routing information between systems.

The RLI helps to prioritize the routes based on the policy as described by the user, which helps to created the BGP update message having routes in the desired order. The feature defines 17 unique priority queues per peer-group, with increasing order of priority.

Test Coverage of the RLI

The test coverage of the RLI includes three main areas

- Route Prioritization as per the configured policy under BGP stanza. Here we test whether the routes are placed in the appropriate priority queues or not
- Priority Overriding – Here we are having user defined policy options, which will provide granular control on which route to put to which queue. It basically can override the policy configured under BGP stanza.
- Scheduling of the priority queues. Here we test whether the routes are taken from the seventeen priority queues as per the priority and created the BGP Update Message. We will also cover any starvation with respect to different priority levels.

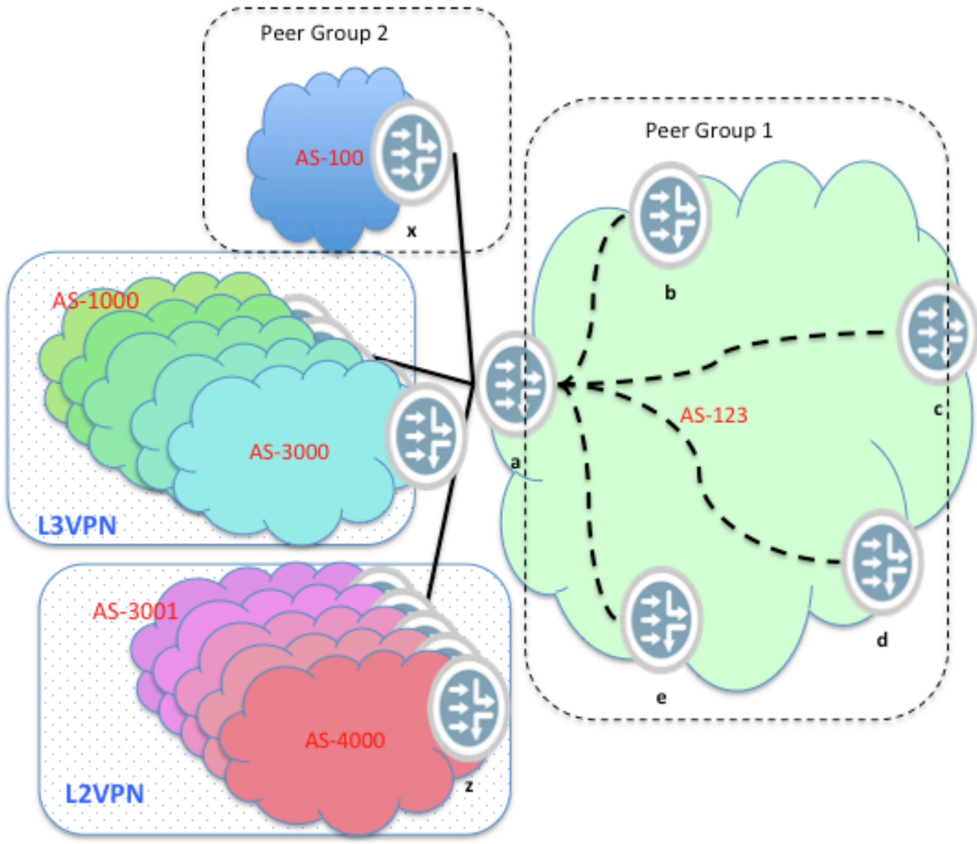
As per the current design the feature is having zero control on how the different priority queues are filled. That is the feature honors the priority for scheduling only when the different priority queues are filled with routes.

Challenges for testing

Major challenges of the feature testing are as follows:

- The feature will be able to verify on a scaled environment with various BGP supported NLRI's
- The queues will be serviced very fast, a tester wont even able to see the routes in the queue
- To see the routes in the queues, we should have congestion happening in the peer routers.
- Even-though the number of priority queues are same for a peer-group, depending upon the congestion and the capabilities of the peer routers, micro-groups can form which creates synced and not-in-sync peers

Consider the following topology



Lets assume that PE router “a” has four ibgp peers in PeerGroup1, and have a router in the peer-group2 for the internet routes, L3VPN cloud with 2000 customers and L2VPN cloud with 1000 customers.

The BGP route prioritization output of a single neighbor of the PE router “a” within the peer-group-1, is having 2000 vrf table information, 1000 l2vpn table information, if say 1000 CE supports IPv6 as well, so we will be having 1000 6vpe tables. Each of these table information is having priority information for 17 priority queues. Hence we have a total of 4000 table to analyze and 68000 priority queue to check. Manually configuring this scale configuration and verifying the output is not a feasible solution.

The following information is displayed for each of the 4000 tables

```
Peer: 65.1.1.1+63883 AS 65500  Local: 65.2.2.2+179 AS 65500
  Output Queue[0]: 49750        (inet.0, inet-unicast)
    Priority 1 :   0
    Priority 2 :   0
    Priority 3 :   0
    Priority 4 :   49750
    Priority 5 :   0
    Priority 6 :   0
    Priority 7 :   0
    Priority 8 :   0
    Priority 9 :   0
    Priority 10:   0
    Priority 11:   0
    Priority 12:   0
    Priority 13:   0
    Priority 14:   0
    Priority 15:   0
    Priority 16:   0
    Expedited  :   0
```

Hence for testing manually the tester has to do

- Create scale configuration of various BGP services
- Have to poll the output of the priority queues at desired intervals
- Have to analyze the desired table from an output info spanned over lakhs of lines
- Have to check the BGP Update message for the order in which routes are packed

Even-though we can test the feature with above verification with the help of CLI show commands and packet analysis, it is hard to do that and very hard to find the behavior change across images.

Mini Project Summary:

Verify the Order of Prioritization for the BGP Routes from pcap files

Mini Project Description:

BGP Carries route information for various family of services. Once the routes are prioritized, the BGP update message is created honoring the priorities of the family and their routes. Tcpdump is enabled on the peer router, before the BGP session is setup, and packets are captured for the duration BGP peering is Established and the peer learns the routes.

In the given pcap file, filter out the BGP messages from a given source to destination, filter out BGP update messages and list out their timestamp, family info and route information.

For example:

User1: ~ user1$ *python bgp_prio.py –file bgpinfo.pcap –src '1.1.1.1' –dst '2.2.2.2'*

TimeStamp	**Family**	**Prefix Start**	**Prefix End**
496.990055	IPv6-Labeled Unicast	2000:2:3:2362::/64	2000:2:3:24ad::/64
497.199888	IPv6-Labeled Unicast	2000:2:3:24ae::/64	2000:2:3:25fa::/64
636.479861	IPv6-Labeled VPN Unicast	2002:61:1:59::/64	2002:61:1:59::/64
646.889723	IPv4-Labeled VPN Unicast	67.1.136.0	67.1.136.0
646.889723	IPv6-Labeled VPN Unicast	2002:67:1:e::/64	2002:67:1:e::/64

Once the above data is available, plot the timestamp versus family.

Input Data: pcap file will be given