Advanced New Developments in BGP

Josef Ungerman
Advanced New Developments in BGP

• High Availability
• Other Features
• Tips
High Availability
High Availability

- Non-Stop Routing
- Next Hop Tracking
- Fast Session Deactivation
- Event Driven Route Origination
- MRAI
- BGP BFD Support
- BGP PIC Overview
Non-Stop Routing

Graceful Restart Review

- GR capability is negotiated on session initiation
- When A restarts, it opens a new TCP session to B, using the same router ID
- B interprets this as a restart, and closes the old TCP session
Non-Stop Routing

Graceful Restart Review

- B transmits updates containing its BGP table (it’s local RIB out)
- A goes into read only mode, and does not run the bestpath calculations until its B has finished sending updates
- When B has finished sending updates, it sends an end of RIB marker, which is an update with an empty withdrawn NLRI TLV
Non-Stop Routing

Graceful Restart Review

- When A receives the end of RIB marker, it runs bestpath, and installs the best routes in the routing table.
- After the local routing table is updated, BGP notifies CEF.
- CEF then updates the forwarding tables, and removes all information marked as stale.
Non-Stop Routing

Graceful Restart Review

- Use the `bgp graceful-restart` command under the `router bgp` configuration mode to enable graceful restart

- `show ip bgp neighbors` can be used to verify graceful restart is operational

```
router bgp 65000
bgp graceful-restart

router bgp 65501
bgp graceful-restart

router#show ip bgp neighbors x.x.x.x
....
Neighbor capabilities:
....
Graceful Restart Capability: advertised and received
Remote Restart timer is 120 seconds
Address families preserved by peer:
IPv4 Unicast, IPv4 Multicast
```
Non-Stop Routing

Graceful Restart Review

• Graceful Restart is now supported for IPv6
• Graceful Restart is now configurable per peer

For the entire process

```
bgp graceful-restart
```

For a single peer

```
ha-mode graceful-restart
```

Within a peer template only

Peer must inherit per neighbor graceful restart from peer template
Non-Stop Routing

Graceful Restart Review

- Graceful restart doesn’t work well in all situations....
- At an eBGP edge, you must count on the “other” AS for support
- This isn’t always possible, especially for a provider edge
Non-Stop Routing

Benefits

• Non-Stop Routing (NSR) fills this gap

• If an eBGP peer is not capable of supporting GR, NSR can be used instead
Non-Stop Routing

Operation

• Each change in peering state is copied to both the active and standby route processors at the same time

• If one route processor fails in the PE, the other route process picks up the session

• The eBGP peer doesn’t see the session change state
Non-Stop Routing

Configuration

```
router bgp 65000
  no synchronization
  bgp log-neighbor-changes
  bgp graceful-restart restart-time 120
  bgp graceful-restart stalepath-time 360
  bgp graceful-restart
  neighbor 10.1.1.1 remote-as 65001
  neighbor 10.1.1.1 update-source Loopback0
  no auto-summary
  !
  address-family vpnv4
  neighbor 10.1.1.1 activate
  neighbor 10.1.1.1 send-community both
      exit-address-family
  !
  address-family ipv4 vrf ce-1
  neighbor 10.3.3.3 remote-as 65001
  neighbor 10.3.3.3 ha-mode sso
  neighbor 10.3.3.3 activate
  neighbor 10.3.3.3 as-override
  no auto-summary
  no synchronization
  exit-address-family
```
Non-Stop Routing

• Support
  12.2(28)SB – eBGP peers

• Assumes you have SSO available and configured

• IOS XR 3.9
Non-Stop Routing
Management and Troubleshooting

• debug ip bgp sso {events|transactions} [detail]
  Events: prints BGP SSO events to the console
  Transactions: prints BGP transactions between the active and standby RPs to the console

• debug ip tcp ha {events|transactions} [detail]
  Events: prints TCP SSO events to the console
  Transactions: prints TCP transactions between the active and standby RPs to the console
Non-Stop Routing
Management and Troubleshooting

- `show ip bgp vpnv4 all sso summary`
  Shows the number of BGP neighbors that support NSF

- `show ip bgp vpnv4 summary`
  Shows whether or not SSO support is enabled

- `show tcp ha connections`
  Shows connection ID to TCP mapping
  Shows connections which have SSO enabled
Nexthop Tracking

BGP Scanner Review

• The BGP scanner plays a key role in convergence
  Two types of scans
    Full or Table: every 60 seconds
    Import: every 15 seconds
  The full scan performs multiple housekeeping tasks
    Validates nexthop reachability
    Validates bestpath selection
    Handles route redistribution and network statements
    Handles conditional advertisement
    Handles route dampening
    Performs a BGP database cleanup

• The import scanner imports VPNv4 routes into the corrects VRFs
**Nexthop Tracking**

**BGP Scanner Review**

- **Nexthop Checks**
  
  If the route to the nexthop for any route changes....

  BGP must run the bestpath algorithm against the route

  Ensures we aren’t black holing traffic in the network, etc.

- **Periodic polling slows convergence way down....**

  What we need to do is make this event driven
Nexthop Tracking

Address Tracking Filter (ATF)

- ATF is a middle man between the RIB and RIB clients
  
  BGP, OSPF, EIGRP, etc. are all clients of the RIB

- A client tells ATF what prefixes it is interested in

- ATF tracks each prefix

  Notify the client when the route to a registered prefix changes

  Client is responsible for taking action based on ATF notification

  Provides a scalable event driven model for dealing with RIB changes
Nexthop Tracking

Address Tracking Filter (ATF)

- BGP tells ATF to let us know about any changes to 10.1.1.3

- ATF filters out any changes for 10.1.1.1/32, 10.1.1.2/32, and 10.1.1.4/32

- Changes to 10.1.1.3/32 and 10.1.1.5/32 are passed along to BGP
**Nexthop Tracking**

**BGP Reaction to Notification**

- Once an ATF notification is received BGP waits five seconds before triggering NHT scan
  
  `bgp nexthop trigger delay <0-100>`
  
  May lower default value as we gain experience

- Dampening is used to reduce frequency of triggered scans

- The full scanner still runs every 60 seconds
  
  Does not check nexthops if NHT is enabled
Nexthop Tracking
Managing and Troubleshooting

- `show ip bgp internal`

  When the last NHT event occurred
  Time until the next NHT may occur (dampening information)

- `show ip bgp attr next-hop ribfilter`

  Shows current address tracking filter

- `debug ip bgp events nexthop`

  Shows nexthop tracking events

- `debug ip bgp rib-filter`

  Shows insertions and deletions from the address tracking filter by BGP
Nexthop Tracking
Managing and Troubleshooting

- `bgp nexthop trigger enable`
  
  On by default, can be turned off using “no” form of this command

- `bgp nexthop trigger delay <0-100>`

  Set delay between receiving an event and reacting to it
Fast Session Deactivation

• If BGP can track nexthops using the ATF, why not track the nexthops for peers, as well?
• This would speed convergence up by bringing a peer down faster than the TCP hold timer
• This is what Fast Session Deactivation (FSD) does
Fast Session Deactivation

- Good for multihop peering sessions
- Similar to fast fallover for connected peers

Fast fallover tracks interface state
FSD tracks reachability
Fast Session Deactivation

- **Not recommended for iBGP peers**
  
  Transient changes in IGP forwarding within your AS could cause major convergence events

- **Off by default**
  
  `neighbor x.x.x.x fall-over`

- **Introduced in 12.0(29)S, 12.3(14)T**
Event-Driven Route Origination

- Route origination was also based on a scanner dependant polling model
- Scanner traversed the RIB looking for routes that should be originated
- Traversing the RIB consumes a lot of CPU
- Route origination is now event driven
  - Scanner no longer checks the RIB for routes to redistribute
  - Route redistribution is event driven
  - Network statements are event driven
  - CPU impact of scanner is greatly reduced
- On by default, cannot disable
- Introduced in 12.2(28)S, 12.3(13)T via CSCef51906
MRAI – Min Route Advertisement Interval

“...determines the minimum amount of time that must elapse between an advertisement and/or withdrawal of routes to a particular destination by a BGP speaker to a peer. This rate limiting procedure applies on a per-destination basis, although the value of MinRouteAdvertisementIntervalTimer is set on a per BGP peer basis.”

RFC 4271
Section 9.2.1.1
MRAI – Basics

• MRAI timers are maintained per peer
  iBGP – 5 seconds by default
  eBGP – 30 seconds by default
  `neighbor x.x.x.x advertisement-interval <0-600>`

• Popular misconception that withdraws are not affected

• Pros
  Promotes stability by batching route changes
  Improves update packing in some situations

• Cons
  May *drastically* slow convergence
  Current defaults are too conservative
  One flapping prefix can slow convergence for other prefixes
BFD Support

- BGP can use BFD for fall-over
  
  bfd interval
  
  BFD hello interval
  
  neighbor fall-over bfd
  
  Enables BGP peer fallover based on BFD failover
  
  show bfd neighbors
  
  Indicates BFD neighbors
  
  show ip bgp neighbors
  
  Indicates BGP neighbors
  
  Includes BFD enabled neighbors
Other Features
Other Features

- VPLS Autodiscovery
- 4 Byte AS
- Passive and Dynamic Neighbors
- Per VRF Router ID
- Minimum Hold Time
- Per Neighbor Policy Show
- BTSH
VPLS Autodiscovery

VPLS Review

- IP/MPLS architecture
- Provides a LAN type Ethernet service over an IP/MPLS infrastructure
- Creates virtualized topologies; multiple VPLS domains may work over the same IP/MPLS infrastructure
- Each instance of the service is private, in the sense that they don’t connect to one another
VPLS Autodiscovery

VPLS Review

• Attachment Circuit
  Could be a physical or logical Ethernet port, ATM bridging (RFC1483), FR bridging (RFC1490), even AToM pseudo wire, etc

• Virtual Circuits
  EoMPLS data encapsulation
  Tunnel label is used to reach remote PE
  VC label is used to identify VFI

• VFI: Virtual Forwarding Interface
  Multiple VFI can exist on the same PE box to separate user traffic like VLAN
VPLS Autodiscovery

VPLS Review

- All the CE devices appear to be connected through one virtual switch

- Forwarding of Frames based on Learned MAC addresses

- Uses a Virtual Forwarding Instances (VFI, like VLAN) for customer separation
VPLS Autodiscovery

VPLS Example Configuration

N-PE3
interface Loopback0
  ip address 10.0.0.3 255.255.255.255

! Define VPLS VFI
l2 vfi vpls11 manual
  vpn id 11
  neighbor 10.0.0.1 encapsulation mpls
  neighbor 10.0.0.4 encapsulation mpls

! Attach VFI to VLAN interface
! VLAN ID is local PE significant
interface Vlan11
  xconnect vfi vpls11

! Attachment circuit config
interface GigabitEthernet5/1
  switchport
  switchport trunk encapsulation dot1q
  switchport mode trunk

N-PE4
interface Loopback0
  ip address 10.0.0.4 255.255.255.255

l2 vfi vpls11 manual
  vpn id 11
  neighbor 10.0.0.1 encapsulation mpls
  neighbor 10.0.0.3 encapsulation mpls

interface Vlan11
  xconnect vfi vpls11

! Attachment circuit
interface GigabitEthernet5/1
  switchport
  switchport trunk encapsulation dot1q
  switchport mode trunk
VPLS Autodiscovery

Example Configuration

NPE3#sh mpls 12 vc 11

<table>
<thead>
<tr>
<th>Local intf</th>
<th>Local circuit</th>
<th>Dest address</th>
<th>VC ID</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>VFI vpls11</td>
<td>VFI</td>
<td>10.0.0.1</td>
<td>11</td>
<td>UP</td>
</tr>
<tr>
<td>VFI vpls11</td>
<td>VFI</td>
<td>10.0.0.4</td>
<td>11</td>
<td>UP</td>
</tr>
</tbody>
</table>

NPE3#sh vfi vpls11

Legend: RT=Route-target, S=Split-horizon, Y=Yes, N=No

VFI name: vpls11, state: up, type: multipoint
VPN ID: 11
Local attachment circuits:
 Vlan11
Neighbors connected via pseudowires:
Peer Address  VC ID  S
10.0.0.1      11    Y
10.0.0.4      11    Y

NPE3#sh mac-add vlan 11

Legend: * - primary entry
age - seconds since last seen
n/a - not available

<table>
<thead>
<tr>
<th>vlan</th>
<th>mac address</th>
<th>type</th>
<th>learn</th>
<th>age</th>
<th>ports</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>2222.2211.1111</td>
<td>dynamic</td>
<td>Yes</td>
<td>0</td>
<td>10.0.0.1, 11 &lt; from NPE-1, VC 11</td>
</tr>
<tr>
<td>*</td>
<td>2222.2233.3333</td>
<td>dynamic</td>
<td>Yes</td>
<td>0</td>
<td>G15/1 &lt; from local AC gig 5/1</td>
</tr>
<tr>
<td>11</td>
<td>2222.2244.4444</td>
<td>dynamic</td>
<td>Yes</td>
<td>0</td>
<td>10.0.0.4, 11 &lt; from NPE-4, VC 11</td>
</tr>
</tbody>
</table>
VPLS Autodiscovery

VPLS Review

• How do we prevent loops in a VPLS network?

• Any packets received off a VC can only be sent to an attachment circuit

• Hence, VPLS requires a full mesh of VCs through the provider’s network
VPLS Autodiscovery

Basics

• Data Plane
  Although VPLS simulate multipoint virtual LAN service, the individual VC is still point-to-point EoMPLS. It uses the same data encapsulation as point-to-point EoMPLS

• Control plane Signaling
  Same as EoMPLS, using directed LDP session to exchange VC information

• Auto-discovery of VPN membership
  Reduces VPN configuration and errors associated with configuration.
  Draft-ietf-l2vpn-vpls-ldp does not mandate an auto-discovery protocol. It can be BGP, Radius, DNS, AD based.
  7600 support BGP based auto discovery from 12.2(33)SRB release, based on draft-ietf-l2vpn-signaling-xx.txt
VPLS Autodiscovery

Example Configuration

! BGP configuration (N-PE3 as example)

router bgp 1
no bgp default ipv4-unicast
bgp log-neighbor-changes
neighbor 10.0.0.1 remote-as 1
neighbor 10.0.0.1 update-source Loopback0
neighbor 10.0.0.4 remote-as 1
neighbor 10.0.0.4 update-source Loopback0

! VPLS VFI config
12 vfi vplsl1 autodiscovery
  vpn id 11
  Interface vlan 11
  xconnect vfi vplsl1

! AC config, the same as before
VPLS Autodiscovery

Show Output

NPE3#show ip bgp 12vpn vpls all sum
BGP router identifier 10.0.0.3, local AS number 1
BGP table version is 4, main routing table version 4

<table>
<thead>
<tr>
<th>Neighbor</th>
<th>V</th>
<th>AS</th>
<th>MsgRcvd</th>
<th>MsgSent</th>
<th>TblVer</th>
<th>InQ</th>
<th>OutQ</th>
<th>Up/Down</th>
<th>State/PfxRcd</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.1</td>
<td>4</td>
<td>1</td>
<td>15</td>
<td>16</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>00:11:40</td>
<td>1</td>
</tr>
<tr>
<td>10.0.0.4</td>
<td>4</td>
<td>1</td>
<td>15</td>
<td>15</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>00:11:11</td>
<td>1</td>
</tr>
</tbody>
</table>

NPE3#sh vfi vpls11

Legend: RT=Route-target, S=Split-horizon, Y=Yes, N=No

VFI name: vpls11, state: up, type: multipoint
VPN ID: 11, VPLS-ID: 1:11
RD: 1:11, RT: 1:11
Local attachment circuits:
Vlan11
Neighbors connected via pseudowires:

<table>
<thead>
<tr>
<th>Peer Address</th>
<th>VC ID</th>
<th>Discovered Router ID</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.4</td>
<td>11</td>
<td>10.0.0.4</td>
<td>Y</td>
</tr>
<tr>
<td>10.0.0.1</td>
<td>11</td>
<td>10.0.0.1</td>
<td>Y</td>
</tr>
</tbody>
</table>
VPLS Autodiscovery

Summary

• There is no need to create an explicit list of PEs and associate them with a given VPN

• When a VPLS instance is created by “l2 vfi” command on that PE, the corresponding VPN-id is distributed by that PE via MP iBGP updates and all the other PEs will become aware of it

• The formats for RD are BGP-ASN:VFI-VPN-ID (default), ASN:nn or IP-address:nn

• Each VSI must have an import and export RT. By default, the RT for each VFI will have the same value as the RD.

• After distribution of PW related parameters, the PWs are setup through targeted LDP signaling
4 Byte AS

- RFC 4271 defines an AS number as 2-bytes
- Private AS Numbers = 64512 through 65535
- Public AS Numbers = 1 through 64511
  - 39000+ have already been allocated
  - We will eventually run out of AS numbers
- Need to expand AS size from 2-bytes to 4-bytes
  - 4,294,967,295 AS numbers
  - Cannot have a “flag day” solution
    - On Jan 1, 2010 all BGP speakers must support feature FOO
    - Solution must support a gradual deployment
4 Byte AS

• draft-ietf-idr-as4bytes-12.txt
  “BGP Support for Four-octet AS Number Space”
  Provides 4-byte AS support without a flag day
  AS numbers will be assigned in X.Y syntax

• X.Y syntax
  AS #65,536,005 is a mouthful
  Split the 4-byte value into two 2-byte values
  00000011111101000 : 0000000000000101
  00000011111101000 = 1000
  0000000000000101 = 5
  1000.5 is easier to work with
4 Byte AS

• 4-byte AS support is advertised via BGP capability negotiation
  Speakers who support 4-byte AS are known as NEW speakers
  Those who do not are known as OLD speakers

• New Reserved AS#
  AS_TRANS = AS #23456
  2-byte placeholder for a 4-byte AS number

• Two new attributes, both are “optional transitive”
  NEW_AGGREGATOR
  NEW_ASPATH
4 Byte AS

• Formatting UPDATEs for a NEW speaker
  Encode each AS number in 4-bytes
  AS_PATH and AGGREGATOR are affected

• Formatting UPDATEs for an OLD speaker
  If the AGGREGATOR/ASPATH does not contain a 4-byte AS we are fine
  If it does, substitute AS_TRANS (AS #23456) for each 4-byte AS
  NEW_AGGREGATOR or NEW_ASPATH will contain a 4-byte encoded copy of the attribute if needed
  OLD speaker will blindly pass along NEW_AGGREGATOR and NEW_ASPATH attributes

*** Target for IOS is mid 2008 in 12.2s and various trains ***
4 Byte AS

- Receiving UPDATEs from a NEW speaker
  Decode each AS number as 4-bytes
  AS_PATH and AGGREGATOR are effected

- Receiving UPDATEs from an OLD speaker
  NEW_AGGREGATOR will override AGGREGATOR
  NEW_ASPATH and ASPATH must be merged to form the correct as-path

- Merging NEW_ASPATH and ASPATH
  ASPATH – 275 250 225 23456 23456 200 23456 175
  NEW_ASPATH – 100.1 100.2 200 100.3 175
  Merged as-path – 275 250 225 100.1 100.2 200 100.3 175
4 Byte AS
Operation Example

<table>
<thead>
<tr>
<th>AS</th>
<th>AS_PATH</th>
<th>NEW_ASPATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>100.1</td>
<td>{100.1}</td>
<td></td>
</tr>
<tr>
<td>700</td>
<td>{700,100.1}</td>
<td>{100.2,700,100.1}</td>
</tr>
<tr>
<td>100.2</td>
<td>{23456,700,23456}</td>
<td>{100.2,700,100.1}</td>
</tr>
<tr>
<td>800</td>
<td>{800,23456,700,23456}</td>
<td>{100.2,700,100.1}</td>
</tr>
<tr>
<td>900</td>
<td>{900,800,100.2,700,100.1}</td>
<td>{700,100.1}</td>
</tr>
<tr>
<td>100.3</td>
<td>{900,800,23456,700,23456}</td>
<td>{700,100.1}</td>
</tr>
</tbody>
</table>

Merge AS-PATH
4 Byte AS
Aggregation Example

AS 100.1
10.1.1.1/32

AS 200
10.1.1.2/32

AS 100.2
10.1.1.3/32

10.1.1.0/24
ASPATH: {23456,[23456,200]}
NEW_ASPATH: {100.3,[100.1,200,100.2]}
AGGREGATOR: 23456 1.1.1.1
NEW_AGGREGATOR: 100.3 1.1.1.1

AS 100.3

AS 300

AS 400

• AS 100.3 creates 10.1.1.0/24 aggregate
4 Byte AS

Configuration Example

router bgp 4.4
  bgp log-neighbor-changes
  neighbor 134.0.0.3 remote-as 3.3

R4#sh ip bgp 1.1.1.0
BGP routing table entry for 1.1.1.0/24, version 2
Paths: (1 available, best #1, table default)
Flag: 0x820
  Not advertised to any peer
  3.3 2 1.1
    134.0.0.3 from 134.0.0.3 (134.0.0.3)
      Origin IGP, localpref 100, valid, external, best
R4#sh ip bgp sum
BGP router identifier 134.0.0.4, local AS number 4.4
BGP table version is 2, main routing table version 2
1 network entries using 124 bytes of memory
1 path entries using 52 bytes of memory
2/1 BGP path/bestpath attribute entries using 184 bytes of memory
1 BGP AS-PATH entries using 40 bytes of memory
0 BGP route-map cache entries using 0 bytes of memory
0 BGP filter-list cache entries using 0 bytes of memory
BGP using 400 total bytes of memory
BGP activity 1/0 prefixes, 1/0 paths, scan interval 60 secs
Neighbor      V    AS MsgRcvd MsgSent   TblVer InQ OutQ Up/Down  State/PfxRcd
134.0.0.3     4    3.3 28 27 2 0 0 00:25:33 1
4 Byte AS

Configuration Example

R3#sh ip route | include B
Codes: L - local, C - connected, S - static, R - RIP, M - mobile, B - BGP
B  2.2.2.0 [20/0] via 123.0.0.2, 00:11:01
B  192.0.0.0/24 [20/0] via 123.0.0.2, 00:11:01
R3#sh ip route 192.0.0.0
Routing entry for 192.0.0.0/24
  Known via "bgp 3.3", distance 20, metric 0
  Tag 2, type external
  Redistributing via ospf 1
  Advertised by ospf 1
  Last update from 123.0.0.2 00:12:14 ago
  Routing Descriptor Blocks:
  * 123.0.0.2, from 123.0.0.2, 00:11:09 ago
    Route metric is 0, traffic share count is 1
    AS Hops 1
    Route tag 2

ip as-path access-list 1 permit ^1\.*$
router bgp 1
  neighbor 4.4.4.4 remote-as 1.4
  neighbor 4.4.4.4 route-map foo in

route-map foo permit 10
  match as-path 1

Note that the "." must be escaped from the regular expression with a "\"
TCP – Active vs. Passive Session

R1 opens TCP session to R2

• Active Session
  If the TCP session initiated by R1 is the one used between R1 & R2 then R1 “actively” established the session.

• Passive Session
  For the same scenario R2 “passively” established the session.

• R1 Actively opened the session
• R2 Passively accepted the session
• Can be configured

  neighbor x.x.x.x transport connection-mode [active|passive]
Dynamic Neighbors

Basics

• Passively listens to a prefix (e.g., 192.168.0.0/24) for incoming BGP sessions

• Using a peer-group as a template for the neighbor configuration

• BGP neighbor relationships are created for incoming TCP connections based on the source IP address of the TCP connection
Dynamic Neighbors

Operation

- In a hub and spoke deployment, only the spokes need configuration when they are added

  The hub has to only allow connection requests from the subnet

  The spokes can use addresses from a single subnet

  New routers only require configuration only on one side and that is on the new router
Dynamic Neighbors

Configuration

```plaintext
cisco(config-router)# router bgp 100
  cisco(config-router-router-bgp-100)# bgp listen limit 100
  cisco(config-router-router-bgp-100)# bgp listen range 10.0.0.0/8 peer-group GRP1
  cisco(config-router-router-bgp-100)# bgp listen range 11.0.0.0/8 peer-group GRP1
  cisco(config-router-router-bgp-100)# bgp listen range 192.168/16 peer-group GRP2
  cisco(config-router-router-bgp-100)# neighbor GRP1 peer-group
  cisco(config-router-router-bgp-100)# neighbor GRP1 remote-as 100
  cisco(config-router-router-bgp-100)# neighbor GRP1 alternate-as 101 102 103
  cisco(config-router-router-bgp-100)# neighbor GRP2 peer-group
  cisco(config-router-router-bgp-100)# neighbor GRP2 remote-as 100
  cisco(config-router-router-bgp-100)#
  cisco(config-router-router-bgp-100)# address-family ipv4
  cisco(config-router-router-bgp-100)# neighbor GRP1 activate
  cisco(config-router-router-bgp-100)# neighbor GRP1 route-map IN1 in
  cisco(config-router-router-bgp-100)# neighbor GRP2 activate
  cisco(config-router-router-bgp-100)# neighbor GRP2 route-map IN2 in
  cisco(config-router-router-bgp-100)# exit-address-family
  cisco(config-router-router-bgp-100)#
```

55© 2009 Cisco Systems, Inc. All rights reserved. Cisco Public
Per VRF Router ID

- VRF-VRF peering on router is generally prevented due to same Router ID check
- This feature enables VRF-VRF peering on a router by allowing unique Router ID per VRF
- Automatic generation and Manual configuration supported
Per VRF Router ID

Configuration

• Explicit Configuration

  ```
  router bgp 123
  address-family ipv4 vrf abc
  bgp router-id w.x.y.z
  ```

• Automatic generation

  ```
  Router bgp 123
  bgp router-id vrf auto-assign
  ```
Minimum Hold Time

- BGP Hold Time gets negotiated to minimum of value advertised by either peer in a BGP session
- Routers administrators would like to be able to prevent very small Hold Time values
- CLI command added for lowest value negotiable

neighbor 2.2.2.2 timers 60 180 90

Advertised Hold Time is 180s and minimum accepted in advertisement from peer is 90s
Per Neighbor Policy Show

show ip bgp neighbor policy

- Easy way to display per neighbor configuration parameters
- Example

  show ip bgp neighbors 192.168.1.2 policy
  Neighbor: 192.168.1.2, Address-Family: IPv4 Unicast
  Locally configured policies:
    route-map ROUTE in
  Inherited polices:
    prefix-list NO-MARKETING in
    route-map ROUTE in
    weight 300
    maximum-prefix 10000
BTSH – BGP TTL Security Hack

- R1 and R2 both use BTSH
- Both sides must configure the feature
  ```
  neighbor x.x.x.x ttl-security hops 1
  Valid TTL = 255 - # hops
  ```
- Packets from R2 will have a TTL of 254
- Packets generated by the hacker will have a TTL less than 254
  - Easy to compare the TTL value vs. the 254 threshold and discard spoofed packets
  - Possible to discard packets at the linecard (IOS XR) – hardware drop n on CRS-1
Tips

Old stuff, but...
BGP Scalability

Global config:

ip tcp path-mtu-discovery !!!!
spd headroom 65535
service tcp-keepalives-in
service tcp-keepalives-out
ip tcp selective-ack
ip tcp mss 9000
ip tcp window-size 64000

Interface config:

hold-queue 4096 in !!!!
hold-queue 4096 out !!!!
mtu 9100
not so well known CLI...

**Configuration partitioning**

Router# `show running-config partition ?`
- `ip-as-path` All IP as-path configurations
- `ip-community` All IP community list configurations
- `route-map` All route-map configurations
- `router` All routing configurations

Router# `show running-config partition router bgp 100`

**Configuration rollback**

Router(config)# `archive config` !!! turn on rollback

Router# `sh archive` !!! show rollback checkpoints

<table>
<thead>
<tr>
<th>Archive #</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>disk0:myconfig-1</td>
</tr>
<tr>
<td>2</td>
<td>disk0:myconfig-2</td>
</tr>
<tr>
<td>3</td>
<td>disk0:myconfig-3 &lt;- Most Recent</td>
</tr>
</tbody>
</table>

Router# `config replace disk0:myconfig-3 list` !!! rollback
IOS XR – structured BGP

RP0/0/CPU0# sh run router bgp 1

router bgp 1
  router-id 10.0.0.1
  address-family ipv4 unicast
    scan-time 20
    network 10.1.0.0 mask 255.255.0.0
    redistribute static
  address-family ipv4 multicast
    redistribute isis 1
    network 224.1.0.0 mask 255.255.0.0
neighbor 1.1.1.1
  remote-as 1
  timers 10 30
  address-family ipv4 unicast
    next-hop-self
    route-policy pass-all in
    route-policy pass-all out
  address-family ipv4 multicast
    next-hop-self
    route-policy pass-all in
    route-policy pass-all out
neighbor 2.1.1.1
  remote-as 2
  ebgp-multihop 4
  address-family ipv4 unicast
    max-prefix 1000
  address-family ipv4 multicast
    route-policy pass-all in
    route-policy pass-all out
    route-reflector-client

RP0/0/CPU0(config)# router bgp 1 neigh 1.1.1.1 addr ipv4 uni no next-hop-self
IOS XR – distributed route processors (DRP)

- CRS-1 Single Chassis System
- LCs added to system can be dynamically assigned to any SDR.
- Additional dRPS can be added - in service - to increase control plane scale of any SDR.
- dRPs and LCs can be dynamically reassigned to meet changing service & b/w needs.
- Up to 8 SDRs supported per (multi) chassis.
- Single-system simplicity, with multi-box fault and administrative isolation.

- The 2 RPs are always paired. They run RIB and BGP Manager.
- dRPs may or may not be paired.
- Example – SDR with DRP only for RR.
IOS XR – Process Placement
Distributed BGP example
Conclusion

• Lots of BGP improvements – scalability, security, operability

• Lots of know-how – ~60% marketshare

• Internet BGP core is moving from IOS to IOS XR
  ISP: Cisco 12000 -> CRS-1
  Enterprise: Cisco 7xxx -> ASR1000
Cisco Networkers
Barcelona
Registrujte se
# BGP memory requirements

<table>
<thead>
<tr>
<th>SAFI</th>
<th>IOS (RISC)</th>
<th>IOS (Intel)</th>
<th>XR (RISC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4 Unicast</td>
<td>746 bytes</td>
<td>900 bytes</td>
<td>570 bytes</td>
</tr>
<tr>
<td>IPv4 Unicast (RIB filter)</td>
<td>-</td>
<td>346 bytes</td>
<td>200 bytes</td>
</tr>
<tr>
<td>IPv6 Unicast</td>
<td>860 bytes</td>
<td>1570 bytes</td>
<td>482 bytes</td>
</tr>
<tr>
<td>IPv6 Unicast (RIB filter)</td>
<td>-</td>
<td>-</td>
<td>182 bytes</td>
</tr>
<tr>
<td>VPNv4 Unicast</td>
<td>240 bytes</td>
<td>408 bytes</td>
<td>192 bytes</td>
</tr>
<tr>
<td>VPNv6 Unicast</td>
<td>588 bytes</td>
<td>464 bytes</td>
<td>224 bytes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SAFI</th>
<th>IOS (RISC)</th>
<th>IOS (Intel)</th>
<th>XR (RISC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4 Unicast</td>
<td>2.25 Million</td>
<td>11 Million (8GB)</td>
<td>8.5 Million</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25 Million (16GB)</td>
<td></td>
</tr>
<tr>
<td>IPv6 Unicast</td>
<td>2 Million</td>
<td>2 Million (8GB)</td>
<td>8 Million</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5 Million (16GB)</td>
<td></td>
</tr>
<tr>
<td>VPNv4 Unicast</td>
<td>7 Million</td>
<td>9 Million (8GB)</td>
<td>8 Million</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22 Million (16GB)</td>
<td></td>
</tr>
<tr>
<td>VPNv6 Unicast</td>
<td>2.9 Million</td>
<td>8.5 Million (8GB)</td>
<td>7.5 Million</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20 Million (16GB)</td>
<td></td>
</tr>
</tbody>
</table>