



New BGP Features

ISP/IXP Workshops

New Features

- Most features be in 12.2S and 12.3T/12.4T IOS branches
- Some features will also be in 12.0S and 12.4
- For details on the exact release that introduces feature X please contact your account team
Or look at the Cisco IOS Feature Navigator
(cisco.com/go/fn)

2

Agenda

- **Not so new features**
 - Peer Templates
 - Update Groups
 - Scalability Improvements
 - Named Extended Community Lists
 - Sequenced Extended Community Lists
 - New AFI/SAFI support
- TCP Developments
- BGP Scanner
- OER
- Miscellaneous

3

Not so new features

- “New Features” from last few years
- Many of these were introduced in 12.0(24)S and 12.2(25)S
- A lot of people still don’t know about these

4

BGP Peer Templates

- **Used to group common configurations**
 - Uses peer-group style of syntax
 - Much more flexible than peer-groups
- **Hierarchical policy configuration mechanism**
 - A peer-template may be used to provide policy configurations to an individual neighbor, a peer-group or another peer-template
 - The more specific user takes precedence if policy overlaps
 - individual neighbor → peer-group → peer-template

5

BGP Peer Templates

- First appeared in 12.0(24)S and 12.2(25)S
 - Integrated in 12.3T, now in 12.4
- Two types of templates
- **Session Template**
 - Can inherit from one session-template
 - Used to configure parameters which are independent of the AFI (address-family-identifier)
 - e.g. remote-as, ebgp-multihop, passwords, etc
- **Peer/policy Template**
 - Can inherit from multiple peer/policy templates
 - Used to configure AFI dependant parameters
 - Filters, next-hop-self, route-reflector-client, etc

6

Session Template

```
router bgp 100
!
template peer-session all-sessions
!
version 4
timers 10 30
exit-peer-session
!
no synchronization
bgp log-neighbor-changes
!
template peer-session iBGP-session
!
remote-as 100
password 7
022F021B12091A6148480A0B1C07064B180C23
38642C2627281D
description iBGP peer
update-source Loopback0
inherit peer-session all-sessions
exit-peer-session
!
template peer-session eBGP-session
!
description eBGP peer
ebgp-multihop 2
inherit peer-session all-sessions
exit-peer-session
!
neighbor 1.1.1.1 inherit peer-session iBGP-session
neighbor 1.1.1.2 inherit peer-session iBGP-session
neighbor 1.1.1.3 inherit peer-session iBGP-session
neighbor 10.1.1.1 remote-as 1442
neighbor 10.1.1.1 inherit peer-session eBGP-session
neighbor 10.1.1.2 remote-as 6445
neighbor 10.1.1.2 inherit peer-session eBGP-session
no auto-summary
!
```

- 1.1.1.1 → 1.1.1.3 are configured with commands from **all-sessions** and **iBGP-session**
- 10.1.1.1 → 10.1.1.2 are configured with commands from **all-sessions** and **eBGP-session**

Policy Template

```
router bgp 100
!
template peer-policy all-peers
!
prefix-list deny-martians in
prefix-list deny-martians out
exit-peer-policy
!
template peer-policy external-policy
!
remove-private-as
maximum-prefix 1000
inherit peer-policy all-peers 10
exit-peer-policy
!
template peer-policy full-routes-customer
!
route-map full-routes out
inherit peer-policy external-policy 10
exit-peer-policy
!
neighbor 1.1.1.1 inherit peer-policy internal-policy
neighbor 1.1.1.2 inherit peer-policy RRC
neighbor 10.1.1.1 inherit peer-policy full-routes-customer
neighbor 10.1.1.2 inherit peer-policy partial-routes-customer
!
```

```
!
template peer-policy partial-routes-customer
!
route-map partial-routes out
inherit peer-policy external-policy 10
exit-peer-policy
!
template peer-policy internal-policy
!
send-community
inherit peer-policy all-peers 10
exit-peer-policy
!
template peer-policy RRC
!
route-reflector-client
inherit peer-policy internal-policy 10
exit-peer-policy
!
```

Policy Template

```
!
template peer-policy foo
!
filter-list 100 out
prefix-list foo-filter out
inherit peer-policy all-peers 10
exit-peer-policy
!
template peer-policy bar
!
prefix-list bar-filter out
exit-peer-policy
!
template peer-policy seq_example
!
inherit peer-policy bar 20
inherit peer-policy foo 10
exit-peer-policy
!
neighbor 10.1.1.3 remote-as 200
neighbor 10.1.1.3 inherit peer-policy seq_example
!
```

```
Router#show ip bgp neighbors 10.1.1.3 policy
Neighbor: 10.1.1.3, Address-Family: IPv4
Unicast
Inherited policies:
prefix-list deny-martians in
prefix-list bar-filter out
filter-list 100 out
Router#
```

- A policy template can inherit from multiple templates
- Seq # determines priority if overlapping policies
Higher seq # has priority

BGP Update Groups

- First appeared in 12.0(24)S and 12.2(25)S
Integrated in 12.3T, now in 12.4
- The Problem: peer-groups help BGP scale but customers do not always use peer-groups, especially with eBGP peers
- The Solution: treat peers with a common outbound policy as if they are in a peer-group

BGP Update Groups

- Peers with a common outbound policy are placed into an update-group
- Reduce CPU cycles
BGP builds updates for one member of the update-group
Updates are then replicated to the other members of the update-group
- Same benefit of configuring peer-groups but without the configuration hassle
- Peer-groups may still be used
Reduces config size
No longer makes a difference in convergence/scalability

BGP Update Groups

- What "neighbor" commands determine a common outbound policy?
Outbound Filters (route-maps, as-path ACLs, etc)
Internal vs. External peer
min-advertisement-interval
ORF (Outbound Route Filtering)
route-reflector-client
next-hop-self
etc...
- "neighbor x.x.x.x default-originate" is an exception
We generate this default on a per-peer basis
Can therefore be ignored for update-group selection
- Inbound policy does not matter

BGP Update Groups

- Example

```
router bgp 100
neighbor 10.1.1.1 remote-as 200
neighbor 10.1.1.1 route-map full-routes out
-
neighbor 10.1.1.30 remote-as 3453
neighbor 10.1.1.30 route-map full-routes out
neighbor 10.2.1.1 remote-as 25332
neighbor 10.2.1.1 route-map customer-routes out
-
neighbor 10.2.1.5 remote-as 6344
neighbor 10.2.1.5 route-map customer-routes out
```

13

BGP Update Groups

- “full-routes” peers are in one update-group
- “customer-routes” peers are in another
- New command: show ip bgp replication
- Displays summary of each update-group
 - # of members
 - # of updates formatted (MsgFmt) and replicated (MsgRepl)

```
Router#show ip bgp replication
BGP Total Messages Formatted/Enqueued : 0/0
```

Index	Type	Members	Leader	MsgFmt	MsgRepl	Csize	Qsize
1	external	30	10.1.1.1	0	0	0	0
2	external	5	10.2.1.1	0	0	0	0

14

BGP Update Groups

- show ip bgp update-group
- Peers with “route-map customer-routes out” are in update-group #2

```
Router#show ip bgp update-group 10.2.1.1
BGP version 4 update-group 2, external, Address Family: IPv4 Unicast
BGP Update version : 0, messages 0/0
Route map for outgoing advertisements is customer-routes
Update messages formatted 0, replicated 0
Number of NLRI's in the update sent: max 0, min 0
Minimum time between advertisement runs is 30 seconds
Has 5 members (* indicates the members currently being sent updates):
10.2.1.1 10.2.1.2 10.2.1.3 10.2.1.4
10.2.1.5
```

15

Scalability

- Bootup convergence and “clear ip bgp *” are the biggest challenges
 - Must converge all of our peers from scratch
 - BGP has to build and transmit a large amount of data
- Multiple ways to improve convergence and scalability
- “ip tcp path-mtu-discovery”
 - Forces TCP to optimize its MSS (max segment size)
 - Reduces TCP overhead dramatically
 - Turn this on to improve scalability
- Interface input queue drops
 - TCP acks can arrive in waves
 - Dropping a TCP ack is costly
 - If you are getting these drops, increase the size of your interface input queues

16

Scalability

- Many incremental changes to BGP algorithms to improve convergence
- Most are related to building and replicating updates as efficiently as possible
- Some are related to reducing BGP transient memory usage
- Others involve improving BGP → TCP interaction

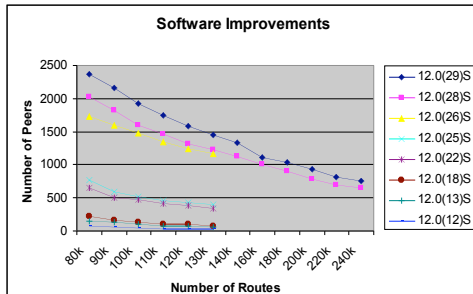
17

Scalability

- “How many peers” graph
- Displays the number of peers we can converge in 10 minutes (Y-axis) assuming we are advertising X-axis number of routes to each peer

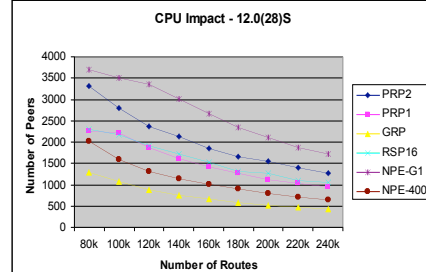
18

Scalability



19

Scalability



- CPU speed plays a big role

20

Named Extended Community Lists

- Named policies are easier to manage than numbered policies
- Support for named extended communities

```
ip extcommunity-list standard AS_100_list permit rt 100:100
ip extcommunity-list expanded AS_2XX_list permit _2[0-9][0-9]_
ip extcommunity-list expanded AS_2XX_list deny .*
```

21

Sequenced Extended Community Lists

- Named and numbered extcommunity-list entries can now have a sequence number
- Allows user to add a statement in a specific location or remove a specific statement
- Example:

```
R1(config)#ip extcommunity-list 44
R1(config-extcomm-list)#10 permit rt 3:3
R1(config-extcomm-list)#20 permit rt 3:10
R1(config-extcomm-list)#30 permit rt 4:4
```

22

Sequenced Extended Community Lists

- Displayed without sequence #s for backwards compatibility

```
R1#sh run | include list 44
ip extcommunity-list 44 permit rt 3:3
ip extcommunity-list 44 permit rt 3:10
ip extcommunity-list 44 permit rt 4:4
```

- The #s are still stored in memory

```
R1#sh ip extcommunity-list 44
Standard extended community-list 44
 10 permit RT:3:3
Standard extended community-list 44
 20 permit RT:3:10
Standard extended community-list 44
 30 permit RT:4:4
```

23

Additional AFI/SAFI Support

- IPv6 VPNs
- IPv6 Multicast
- Multicast VPNs
- For more details refer to the IPv6 and Multicast Cisco Networkers sessions and the Cisco website (cco.cisco.com)

24

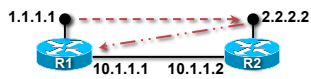
Agenda

- Not so new features
- **TCP Developments**
 - Source/Destination Address Matching
 - Active vs Passive Sessions
 - BTDH - BGP TTL Security Hack
 - TCP PMTU - TCP Path MTU Discovery
- BGP Scanner
- OER
- Miscellaneous

Source/Destination Address Matching

- Both peers must now agree on peering addresses
- IP Addresses
 - Destination IP is specified via "neighbor x.x.x.x"
 - Source IP is outbound interface by default
 - Source IP may be specified via "neighbor x.x.x.x update-source interface"
- TCP port numbers
 - Destination will be port 179
 - Source port is random for added security

Source/Destination Address Matching

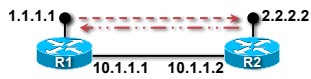


- Both sides must agree on source/destination addresses
- R1 to R2 connection
 - neighbor 2.2.2.2 remote-as 100
 - neighbor 2.2.2.2 update-source loopback 0
- R2 to R1 connection
 - neighbor 10.1.1.1 remote-as 100
 - neighbor 10.1.1.1 update-source loopback 0
- R1 and R2 do not agree on what addresses to use
 - BGP will tear down the TCP session due to the conflict
 - Points out configuration problems and adds some security

Source/Destination Address Matching

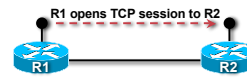
- R2 attempts to open a session to R1
 - BGP: 10.1.1.1 open active, local address 2.2.2.2
- R1 denies the session because of the address mismatch
- "debug ip bgp" on R1 shows
 - BGP: 2.2.2.2 passive open to 10.1.1.1
 - BGP: 2.2.2.2 passive open failed - 10.1.1.1 is not update-source Loopback0's address (1.1.1.1)

Source/Destination Address Matching



- R1 to R2 connection
 - neighbor 2.2.2.2 remote-as 100
 - neighbor 2.2.2.2 update-source loopback 0
- R2 to R1 connection
 - neighbor 1.1.1.1 remote-as 100
 - neighbor 1.1.1.1 update-source loopback 0
- Routers agree on source/destination address
 - BGP will accept this TCP session

TCP – Active vs. Passive Session



- 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 on R2:

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

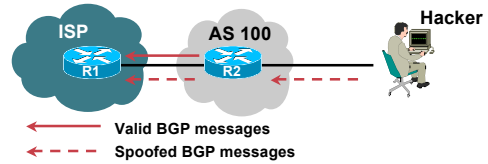
TCP – Active vs. Passive Session

- Use “show ip bgp neighbor” on R1 to determine if a router actively or passively established a session


```
R1#show ip bgp neighbors 2.2.2.2
BGP neighbor is 2.2.2.2, remote AS 200, external link
BGP version 4, remote router ID 2.2.2.2
[snip]
Local host: 1.1.1.1, Local port: 12343
Foreign host: 2.2.2.2, Foreign port: 179
```
- TCP open from R1 to R2's port 179 established the session
- Tells us that R1 actively established the session

31

BTSH – BGP TTL Security Hack



Problem:

- Hackers spoof BGP messages to R1 as if they are R2
- R1 must use MD5 to filter out the bogus messages
- MD5 validation must be done on the RP (Route Processor)

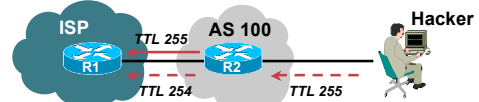
32

BTSH – BGP TTL Security Hack

- Available from 12.0(27)S, 12.2(25)S, 12.3(7)T and 12.4
- Provides a lightweight mechanism to defend against most BGP spoof attacks
 - Does **NOT** replace the need for MD5 authentication!
- Sender sets the TTL to 255
- Receiver checks for a TTL of 254 for directly connected neighbors
 - A lower acceptable TTL value must be configured for multihop neighbors

33

BTSH – BGP TTL Security Hack



- R1 and R2 both use BTSH
- Both sides must configure the feature


```
neighbor x.x.x.x ttl-security 255
```

 May use BTSH instead of ebgp-multihop if you control both ends of the session
- Packets from R2 will have a TTL of 255
- Packets generated by Hackers will have a TTL that is less than 255
 - Easy to compare the TTL value vs. the 255 threshold and discard spoofed packets
 - Discards can be done at the linecard
 - TTL check is much cheaper than MD5

34

BTSH – BGP TTL Security Hack

- Attack scope is reduced to directly connected devices!
- MD5 should still be used to authenticate any message that makes it past BTSH
- BTSH is documented in the Experimental RFC 3682

35

TCP – Security Summary

- Minimal built in security
 - Random source port #s
 - Strict source/destination IP agreement
- TCP's MD5 authentication should be used


```
neighbor x.x.x.x password FOO
```
- MD5 + BTSH (BGP TTL Security Hack) provides protection with minimal CPU cost

36

TCP MSS – Max Segment Size

- **MSS** – Limit on the largest packet that can traverse a TCP session
 - Anything larger must be fragmented & re-assembled at the TCP layer
 - MSS is 536 bytes by default !!!
- 536 bytes is inefficient for Ethernet (MTU of 1500) or POS (MTU of 4470) networks
 - TCP is forced to break large packets into 536 byte chunks
 - Adds overheads
 - Slows BGP convergence and reduces scalability

37

TCP MSS – Max Segment Size

- “ip tcp path-mtu-discovery”
 - MSS = Lowest MTU between destinations
 - minus IP overhead (20 bytes)
 - minus TCP overhead (20 bytes)
 - 1460 bytes for Ethernet network
 - 4430 bytes for POS network
- Will be enabled by default for BGP sessions in the future
- New knob will allow you to enable/disable per peer
 - `[no] neighbor x.x.x.x transport path-mtu-discovery`

38

TCP MSS – Max Segment Size

- BGP Messages
- | | | | |
|----|--------|----|--------|
| KA | Update | KA | Update |
|----|--------|----|--------|
- TCP Packets
- MSS of 536
- | | | | | |
|-----------|--|--|--|--|
| 536 bytes | | | | |
|-----------|--|--|--|--|
- TCP Packets
- MSS of 1460
- | | |
|------------|--|
| 1460 bytes | |
|------------|--|
- BGP KAs (Keepalives) are 19 bytes
 - BGP Updates vary in size up to 4096 bytes
 - The larger the TCP MSS the fewer TCP packets required
 - Fewer packets means less overhead and faster convergence

39

Agenda

- Not so new features
- TCP Developments
- **BGP Scanner**
 - ATF – Address Tracking Feature
 - NHT – Next Hop Tracking
 - Event driven redistribution
- OER
- Miscellaneous

40

BGP Scanner – Overview

- BGP Scanner
- Import scanner runs once every 15 seconds
 - Imports VPNv4 routes into vrfs (2547)
 - `bgp scan-time import X`
- Full scanner run happens every 60 seconds
 - `bgp scan-time X`
 - Lowering this value is not recommended
- Full scan performs multiple housekeeping tasks
 - Validate nexthop reachability
 - Validate bestpath selection
 - Route redistribution and network statements
 - Conditional advertisement
 - Route dampening
 - BGP Database cleanup

41

BGP Scanner – Overview

- CPU spike is normal when scanner runs
 - Is a low priority process
 - Scanner spike shouldn't adversely effect other processes
- Scanning a full table of internet routes is a big job
- “debug ip bgp events” will show you when scanner ran for each address-family

```
BGP: Performing BGP general scanning
BGP(0): scanning IPv4 Unicast routing tables
BGP(IPv4 Unicast): Performing BGP Nexthop scanning for general scan
BGP(0): Future scanner version: 7, current scanner version: 6
BGP(1): scanning IPv6 Unicast routing tables
BGP(IPv6 Unicast): Performing BGP Nexthop scanning for general scan
BGP(1): Future scanner version: 13, current scanner version: 12
BGP(2): scanning VPNv4 Unicast routing tables
BGP(VPNv4 Unicast): Performing BGP Nexthop scanning for general scan
BGP(2): Future scanner version: 13, current scanner version: 12
BGP(4): scanning IPv4 Multicast routing tables
BGP(IPv4 Multicast): Performing BGP Nexthop scanning for general scan
BGP(4): Future scanner version: 13, current scanner version: 12
BGP(5): scanning IPv6 Multicast routing tables
BGP(IPv6 Multicast): Performing BGP Nexthop scanning for general scan
BGP(5): Future scanner version: 13, current scanner version: 12
```

42

ATF – Address Tracking Filter

- Available from 12.0(28)S
(CSCec17043, CSCee70421)
- ATF is a middle man between clients that use the RIB and the FIB
Clients could be BGP, OSPF, EIGRP, etc
- The client tells ATF what prefixes he is interested in
- ATF tells the client when one of these prefixes has a RIB change

BGP NHT – Next Hop Tracking

- Integrated in 12.0(28)S
(CSCec18878, CSCec55381)
Enabled by default
`[no] bgp nexthop trigger enable`
- BGP registers all nexthops with ATF
Hidden command will let you see a list of nexthops
`show ip bgp attr nexthop`
- ATF will let BGP know when a route change occurs for a nexthop
- ATF notification will trigger a lightweight “BGP Scanner” run
Only bestpath will be calculated
None of the other standard stuff that BGP does in scanner will happen

Next Hop Tracking

- BGP will scan the table and recalculate bestpaths
- No longer have to wait as long as 60 seconds for BGP to scan the table and recalculate bestpaths
- Once an ATF notification is received BGP waits 5 seconds before triggering NHT scan
`bgp nexthop trigger delay <0-100>`
May lower default value as we gain experience
- Allows BGP to react quickly to IGP changes
Tuning your IGP for fast convergence is highly recommended

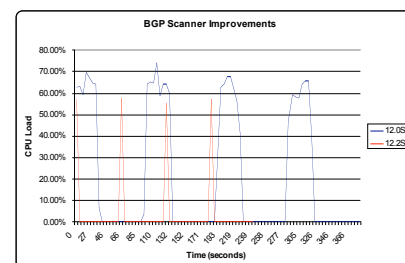
Next Hop Tracking

- Damping library is used to prevent triggered scans from happening too often
“show ip bgp internal” shows when the next scan can run
- New commands
`bgp nexthop trigger enable`
`bgp nexthop trigger delay <0-100>`
`show ip bgp attr next-hop ribfilter`
`debug ip bgp events nexthop`
`debug ip bgp rib-filter`
- Normal BGP scan still happens every 60 seconds
- Normal scanner does not evaluate best path at each net if NHT is enabled

Event Driven Route Origination

- Improvements have been made to reduce CPU impact
Route redistribution is now fully event driven
Network statements are now fully event driven
- Nexthop Tracking (NHT)
NHT detects that our route to one of our BGP nexthops has changed
NHT triggers a lightweight scanner run that only validates nexthop reachability and recalculates bestpaths
Nexthop and bestpath validation no longer happens in scanner every 60 seconds

BGP Scanner



- 7200 with NPE-G1
- 900k routes in the BGP table
- BGP Scanner in 12.2S uses much less CPU

Agenda

- Not so new features
- TCP Developments
- BGP Scanner
- **OER**
 - The Basics
 - BGP's role in OER
- Miscellaneous

49

OER – Optimized Edge Routing

- BGP defines a best path based on a complicated 12 step program
- Shortest AS-PATH is normally the determining factor
- # of ASs in an AS-PATH is a very generic metric
 - Tells us nothing about the number of routers or types of links the traffic will traverse
 - A path with a longer AS-PATH could be faster than a path with a shorter AS-PATH
- AS-PATH prepending and other policies make this picture even more muddy

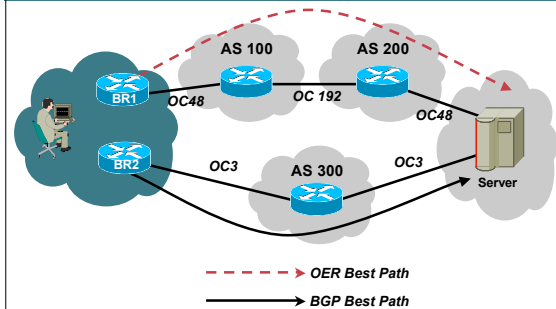
50

OER – Optimized Edge Routing

- OER allows traffic to use the optimal exit point out of a network as opposed to the BGP defined best path
- OER determines this optimal exit point based on information about the actual state of the network (by active and passive network traffic probing)
- The optimal exit is the one giving the best overall performance when trying to communicate with a given prefix

51

OER – Optimized Edge Routing



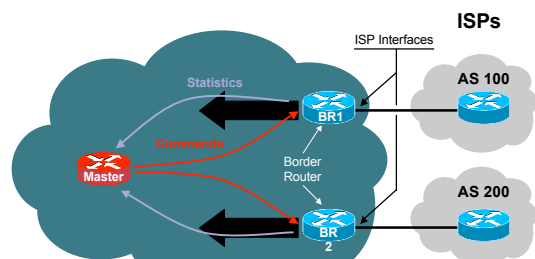
52

OER – How does it work?

- Netflow gathers information to determine delay over various paths
- Netflow data is delivered to a Master OER server
- Server applies user defined policies and rules to determine the optimal exit
- Server changes the BGP configuration of border routers to force traffic out via the optimal exit route-maps and localpref

53

OER – How does it work?



54

OER – Server Settings

- The Master server determines the optimal path by using the Netflow data with user defined policy

Low delay
Low packet loss
Cost Minimization
History
etc.

55

OER – More Information?

- This was OER from 100,000 feet
- Cisco Networkers has an entire session dedicated to OER!
RST-4311

55

Agenda

- Not so new features
- TCP Developments
- BGP Scanner
- OER
- Miscellaneous
 - FSD – Fast Session Deactivation
 - EIGRP PE/CE
 - Restart after max-prefix exceeded
 - Last AS prepend
 - eBGP disable-connected-check
 - RIB Modify

57

FSD

- FSD – Fast Session Deactivation
- Register peers' addresses with ATF
- ATF will let BGP know if there is a change to a peer's address
- If we loose our route to the peer from the RIB, tear down the session
 - No need to wait for the hold timer to expire!

58

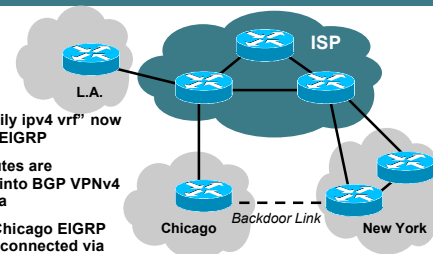
FSD

- Ideal for IBGP peers and multihop eBGP peers
- Can tear down BGP sessions at IGP convergence speed
- Off by default
 - `neighbor x.x.x.x fall-over`

59

EIGRP PE/CE

- "address-family ipv4 vrf" now supported in EIGRP
- EIGRP vrf routes are redistributed into BGP VPNv4 and vice versa
- The LA, NY, Chicago EIGRP networks are connected via 2547 VPNs



60

EIGRP PE/CE

- EIGRP will prefer routes learned via the ISP over the backdoor routes (use of cost-communities)
- All EIGRP metrics are preserved across the ISP backbone!
 - If New York redistributes 10.0.0.0/8 from RIP to EIGRP then LA will see the EIGRP route as an external with the proper metric
 - Accomplished by using BGP extended communities to carry the EIGRP information through the backbone

61

Restart after Max Prefix exceeded

- `neighbor x.x.x.x maximum-prefix 100`
 - Session will be shutdown if peer exceeds limit (100 prefixes)
 - Manual intervention required to re-establish connection
- New "restart" keyword
 - Specify # of minutes to wait before automatically restarting the session
- Do not set the restart timer too low
 - Frequently flapping sessions could result in dampening
 - Give your neighboring operators time to correct the problem
- `neighbor x.x.x.x maximum-prefix 100 restart 30`
 - Session will automatically attempt to re-establish after 30 minutes

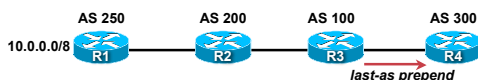
62

Last AS Prepend

- New knob for route-map as-path prepending
 - Only applicable on route-maps applied to neighbor statements
- `set as-path prepend last-as X`
 - Prepends the last-as (leftmost AS in the AS_PATH) X times
- BGP now sanity checks route-map match and set statements
 - R3(config-router)#redist static route-map foo
 - % "foo" used as redistribute static into bgp route-map, set as-path prepend last-as not supported

63

Last AS Prepend

- 
- R3 is configured to last-as prepend towards R4


```
router bgp 100
  neighbor R4 route-map foo out
  route-map foo permit 10
    set as-path prepend last-as 2
```
 - R4 sees the as-path as if R2 prepended


```
R4# show ip bgp 10.0.0.0/8
BGP routing table entry for 10.0.0.0/8, version 41
100 200 200 200 250
20.255.255.2 from 20.255.255.1 (1.1.1.1)
  Origin incomplete, localpref 100, valid, external, best
```

64

eBGP disable-connected-check

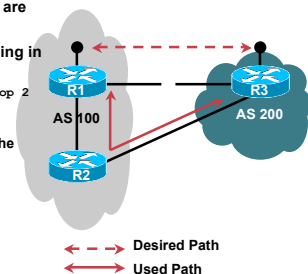
- eBGP peers must meet one of the following criteria
 - Are directly connected which is verified by comparing the eBGP peer's address with our connected subnets
 - Are configured for `ebgp-multihop` which disables the connected subnet check
- Single hop eBGP loopback peering does not fit either rule very well
 - Default TTL (Time To Live) is 1
 - ⇒ `"neighbor x.x.x.x ebgp-multihop 1"` is silently ignored by the parser
 - `"neighbor x.x.x.x ebgp-multihop 2"` must be used here

65

eBGP disable-connected-check

- R1 and R3 are eBGP peers that are loopback peering
- Older code must use the following in R1 and R3


```
neighbor x.x.x.x ebgp-multihop 2
```
- Small security hole
 - If the R1 to R3 link goes down the session could establish via R2



66

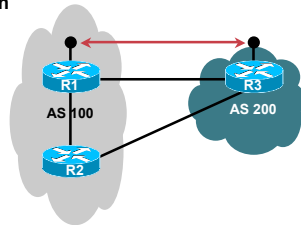
eBGP disable-connected-check

- New code does not need an ebgp-multi-hop statement

Instead use:

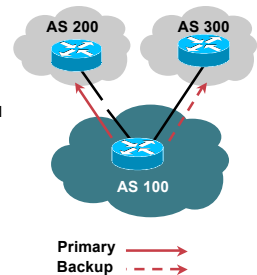
```
neighbor x.x.x.x disable-  
connected-check
```

- TTL is 1
- Session cannot establish via R2
- If R1 to R3 link is down so is the BGP session
- Closes security hole!



RIB Modify

- AS 100 is dual peering
 - AS 200 is primary
 - AS 300 is backup
- Upon AS 200 failure
 - All routes via AS 200 will be deleted
 - Routes via AS 300 will be added
- Brief period where traffic is dropped during transition



RIB Modify

- RIB Modify lets us modify the route in place
- No longer need to do a delete/add
- We modify the AS 200 route with the AS 300 route
- Zero traffic is dropped during the transition!

BGPv4 Soft-Notification

- A NOTIFICATION message resets the BGP session
- The error may apply to only a particular AFI/SAFI
- The #AFI/SAFIs has increased in the recent times
- Affects stability and robustness of BGP Networks

BGPv4 Soft-Notification

- Need a per AFI/SAFI NOTIFICATION that
 - Will not reset the BGP session
 - Will soft-reset the affected AFI/SAFI
 - Has a mechanism to soft-shut/soft-unshut an AFI/SAFI
 - Has a mechanism to synchronize AFI/SAFI states on sender and receiver
 - Would introduce a new Capability

BGPv4 Soft-Notification

- A New BGP Message Type
- No BGP session-reset
- Will soft-reset the affected AFI/SAFI
- Handshaking mechanism to synchronize the AFI/SAFI states between the BGP Speakers sending/receiving the Soft-Notification Message

BGPv4 Soft-Notification

- Updates, update errors and Cease Notifications are per AFI/SAFI
- 70% per AFI/SAFI errors are recoverable
- Remaining 30% could be solved through BGP Update-v2
Changing implementation to encode MP_UNREACH/MP_REACH as the first attribute (Enke's suggestion)

73

Inform vs. Soft-Notification

- **Inform**
 - To signal events or innocuous errors
 - Action taken on receiving an Inform - Logging
- **Soft-Notification**
 - Specifically to signal Soft-Notifications for per-AFI/SAFI errors
 - Action taken on receiving Soft-Notification – AFI/SAFI reset, AFI/SAFI shut or AFI/SAFI unshut
 - Handshaking mechanism to synchronize peer states

74

BGPv4 Soft-Notification – Benefits

- Provides AFI/SAFI robustness and isolation
- New AFI/SAFI deployment leaves the existing AFI/SAFIs unaffected
- Better Network manageability and stability
- New non-routing/routing-related AFI/SAFIs will not affect core Internet routing

75

Agenda

- Not so new features
- TCP Developments
- BGP Scanner
- OER
- Miscellaneous

76



New BGP Features

ISP/IXP Workshops