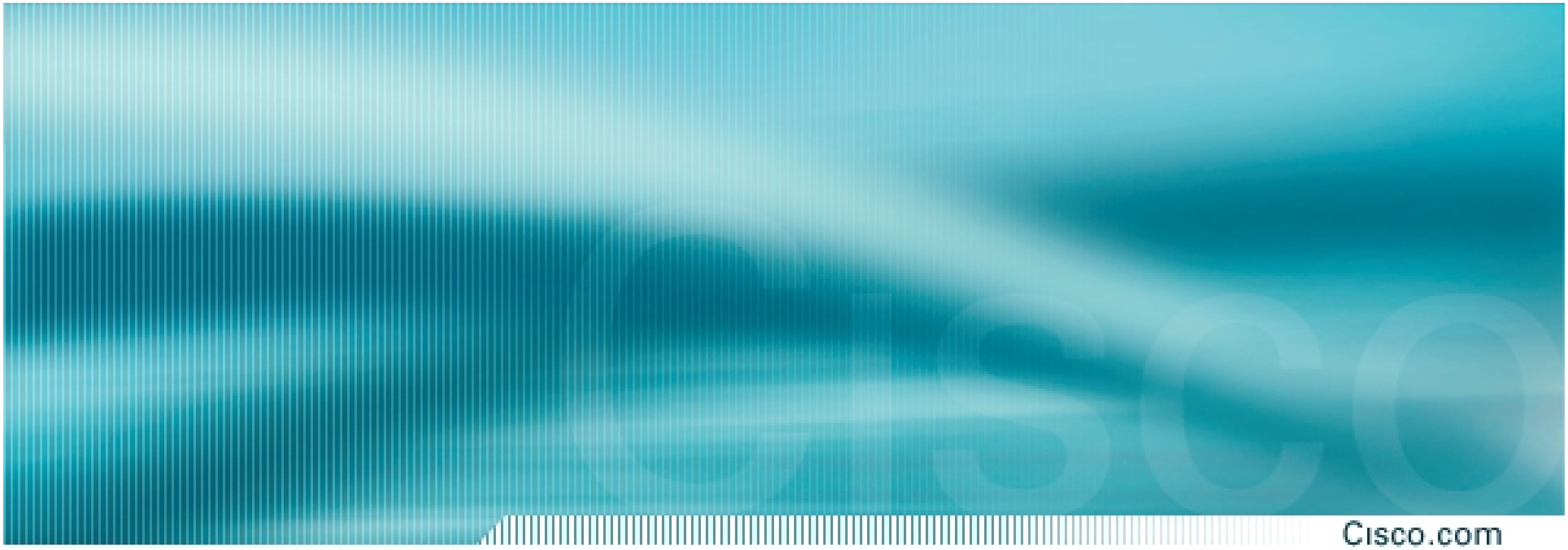


BGP Scalability and Troubleshooting

A little about me

- Speaker: Daniel Walton

  dwalton@cisco.com

- Routing Protocols Deployment & Scalability Team
- Deployment

  Participate at NANOG, Networkers, IETF, RIPE, etc

  Help customers with BGP problems/deployment

- Scalability

  Find convergence bottlenecks

  Find limits on how many peers and routes we can support

Introduction

- My assumptions

  Operational experience with BGP

  Intermediate to advanced knowledge of the protocol

- What can you expect to get from this presentation?

  Discuss scalability problems of the protocol and solutions to work around them

  Ways to improve initial (reboot) convergence

  Learn how to use show commands and debugs to troubleshoot BGP problems

  Go through various real world examples

Agenda

- **Scalability**

  Protocol Issues

  Initial Convergence

- **Troubleshooting**

  Peer establishment

  Missing Routes

  Inconsistent Route Selection

  Loops and Convergence Issues

Protocol Issues - Agenda

- iBGP Full Mesh
- Route Reflectors
- Confederations
- Detection and Propagation of Changes
  - minRouteAdvertisementInterval
  - NEXT_HOP Reachability
  - Route Dampening

iBGP Full Mesh

Cisco.com

> “When a BGP speaker receives an UPDATE message from an internal peer, the receiving BGP speaker shall not re-distribute the routing information contained in that UPDATE message to other internal peers...”

draft-ietf-idr-bgp4-13

Section 9.2.1

iBGP Full Mesh

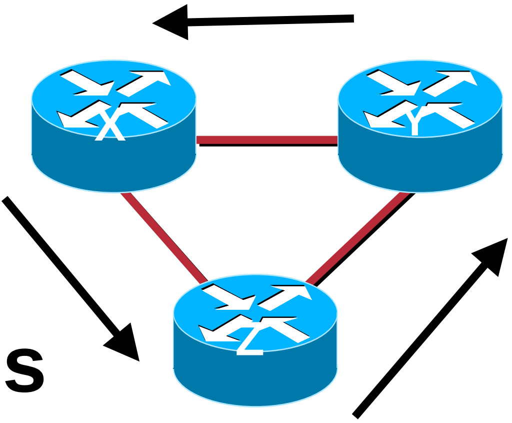
- **Why have this restriction?**

  No mechanism to detect an UPDATE loop exists in iBGP.

- **What may be the consequences of not having a full iBGP mesh?**

  Black holes and routing loops.

  UPDATE loops.



iBGP Full Mesh

Scalability Concerns

- Administration

  Configuration management on increasingly large number of routers.

- Number of TCP Sessions

  Total number of sessions = n(n-1)/2

  Maintaining extreme numbers of TCP sessions creates extra overhead.

- BGP Table Size

  A higher number of neighbors generally translates to a higher number of paths for each route.

  Memory consumption.

Protocol Issues - Agenda

- iBGP Full Mesh
- Route Reflectors
- Confederations
- Detection and Propagation of Changes
    - minRouteAdvertisementInterval
    - NEXT_HOP Reachability
    - Route Dampening

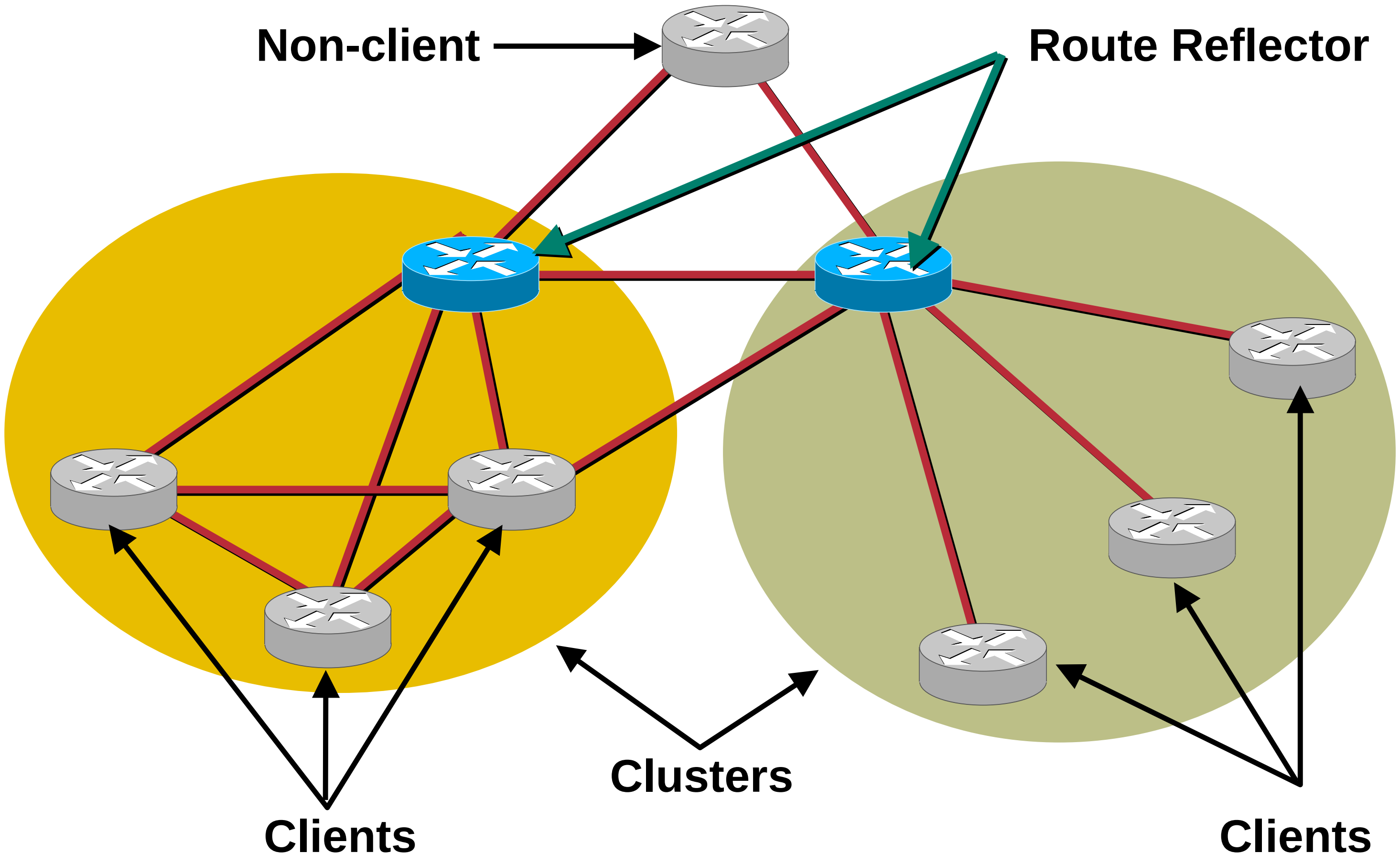
Route Reflectors

- Defined in RFC 2796.
- Allows a router (route reflector – RR) to advertise routes received from an iBGP peer to other iBGP peers.

    Between clients and from clients to non-clients, and vice versa.

- The ORIGINATOR_ID and CLUSTER_LIST attributes are used to perform loop detection.
- Provides a scalable alternative to an iBGP full mesh.

Route Reflectors - Terminology



Lines Represent Both Physical Links and BGP Logical Connections

Route Reflectors

- Only the best path is propagated

  From an eBGP peer, send the path to everyone

  From a RRC, reflect the path to clients and non-clients, send the path to eBGP peers

  From a regular iBGP peer (non-client), reflect the path to RRCs and send the path to eBGP peers

- When a route is reflected the RR appends its ROUTER_ID (or configured *bgp cluster-id*) to the CLUSTER_LIST

Route Reflectors - Deployment

- Divide the network into multiple clusters

  Typical to base cluster assignment on geographic layout

- Each cluster contains at least one RR.

  Clients can peer with RRs in other clusters for redundancy.

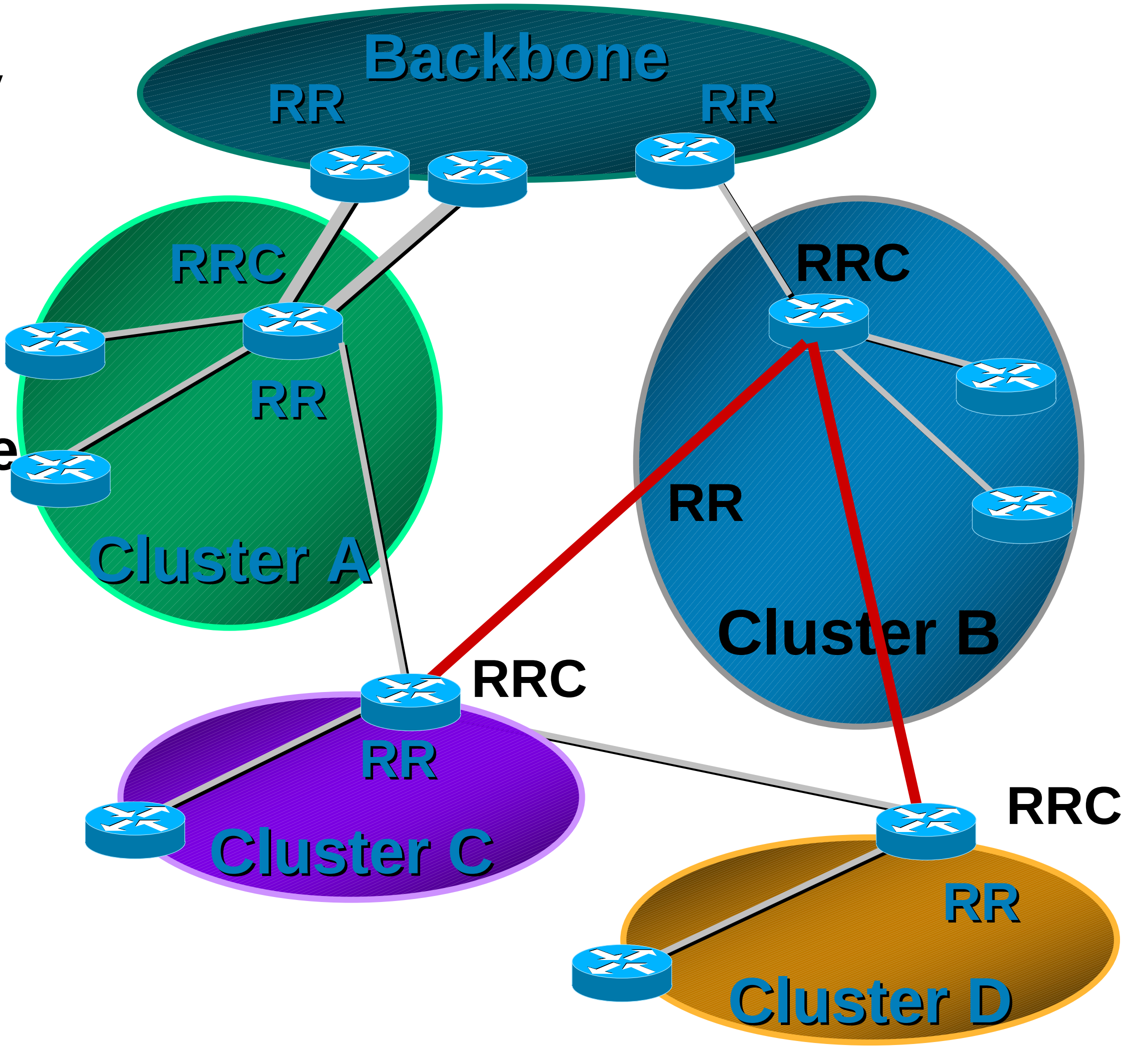
- Top Level RRs are fully meshed via iBGP.

- Still use single IGP — NEXT_HOP unmodified by RR unless via explicit route-map.

Route Reflectors - Deployment

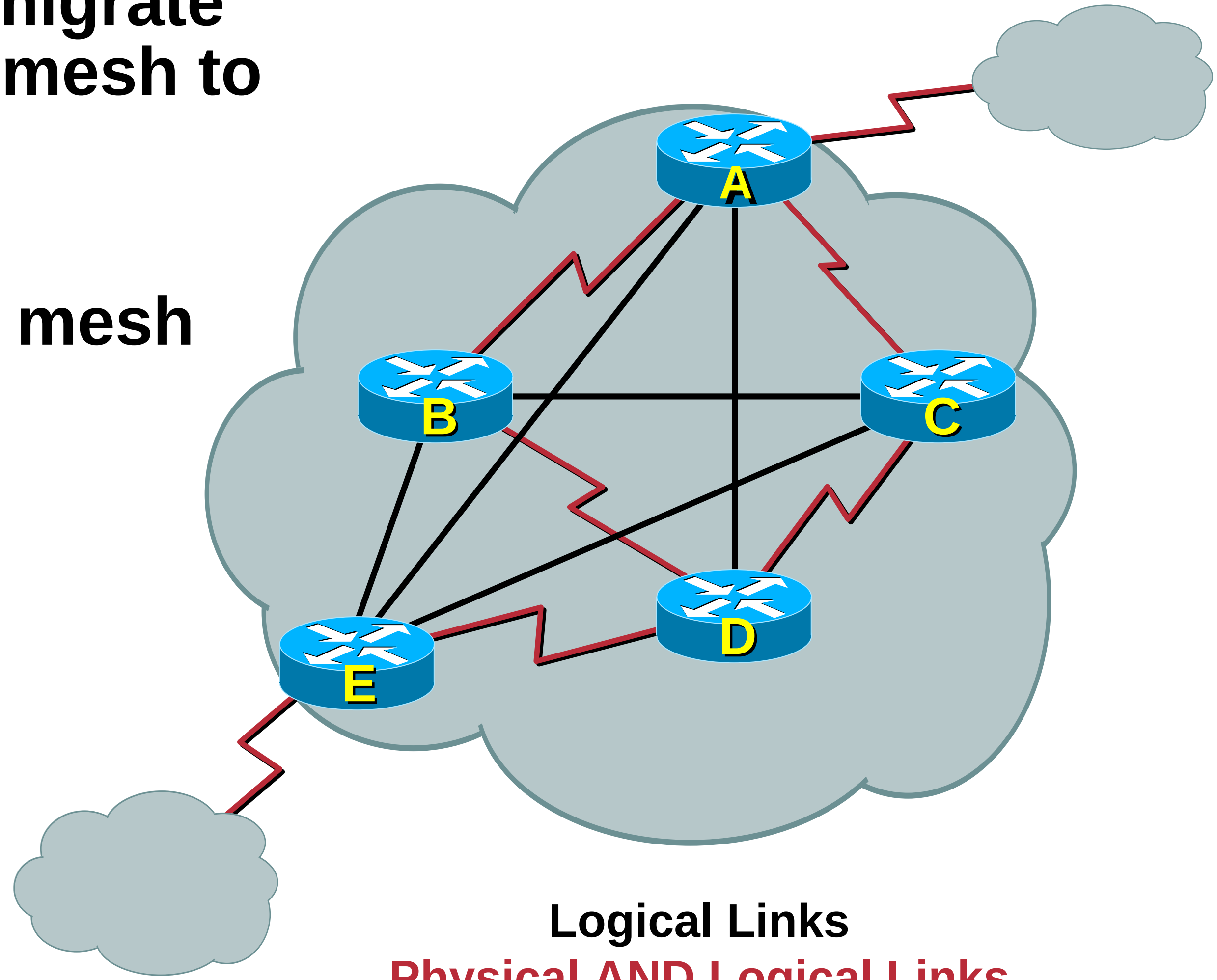
- Follow physical topology
- Do not have to be extremely strict

  Loopback peering is ok
- Do not have a RRC in one region peer with a RR in another
- Do not peer through one RR to get to another
- Routing loops can occur otherwise



Route Reflectors—Migration

- Need to migrate from full mesh to RRs?
- Step 0:
  full iBGP mesh

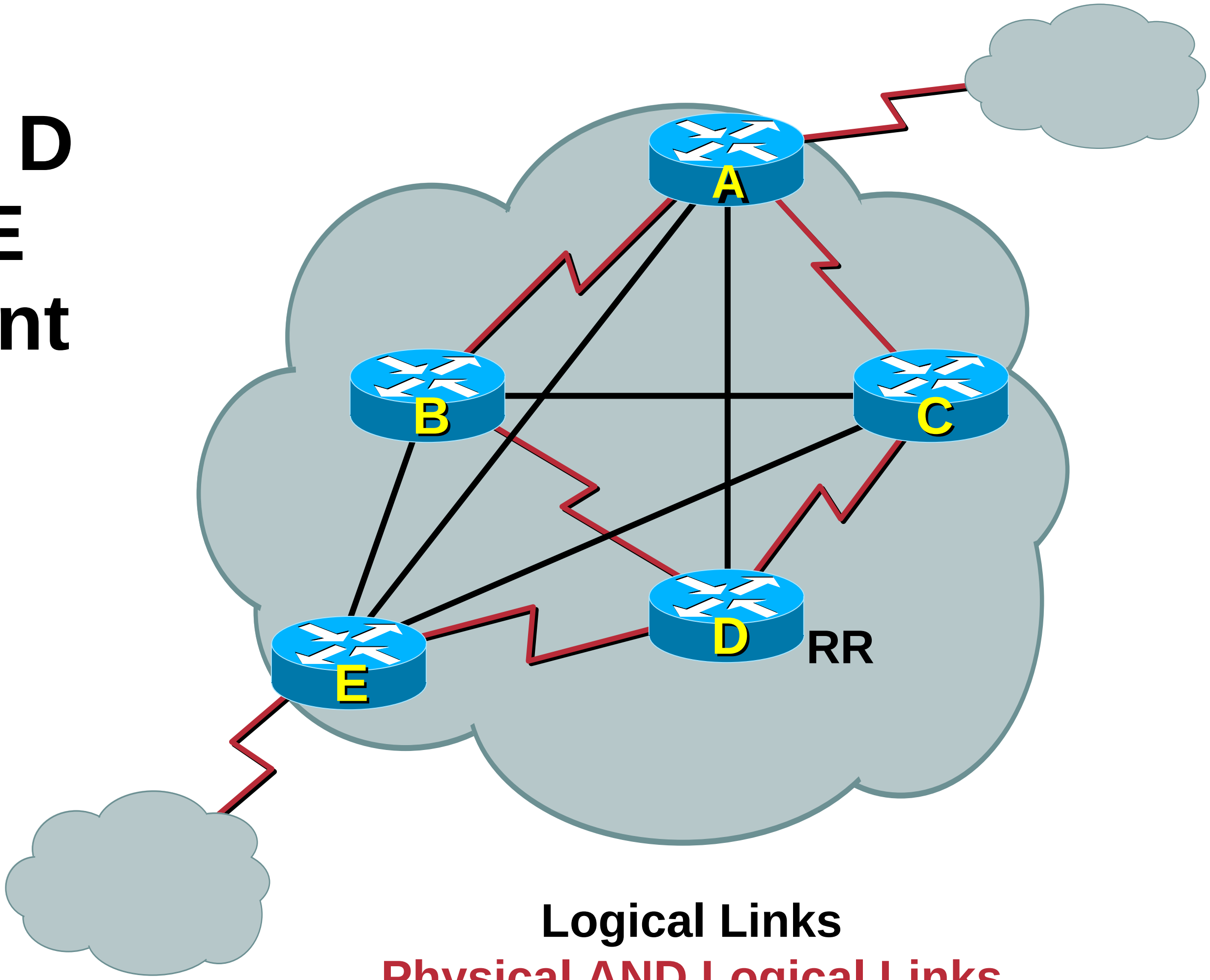


Logical Links

Physical AND Logical Links

Route Reflectors—Migration

- **Step 1: configure D as a RR; E is the client**

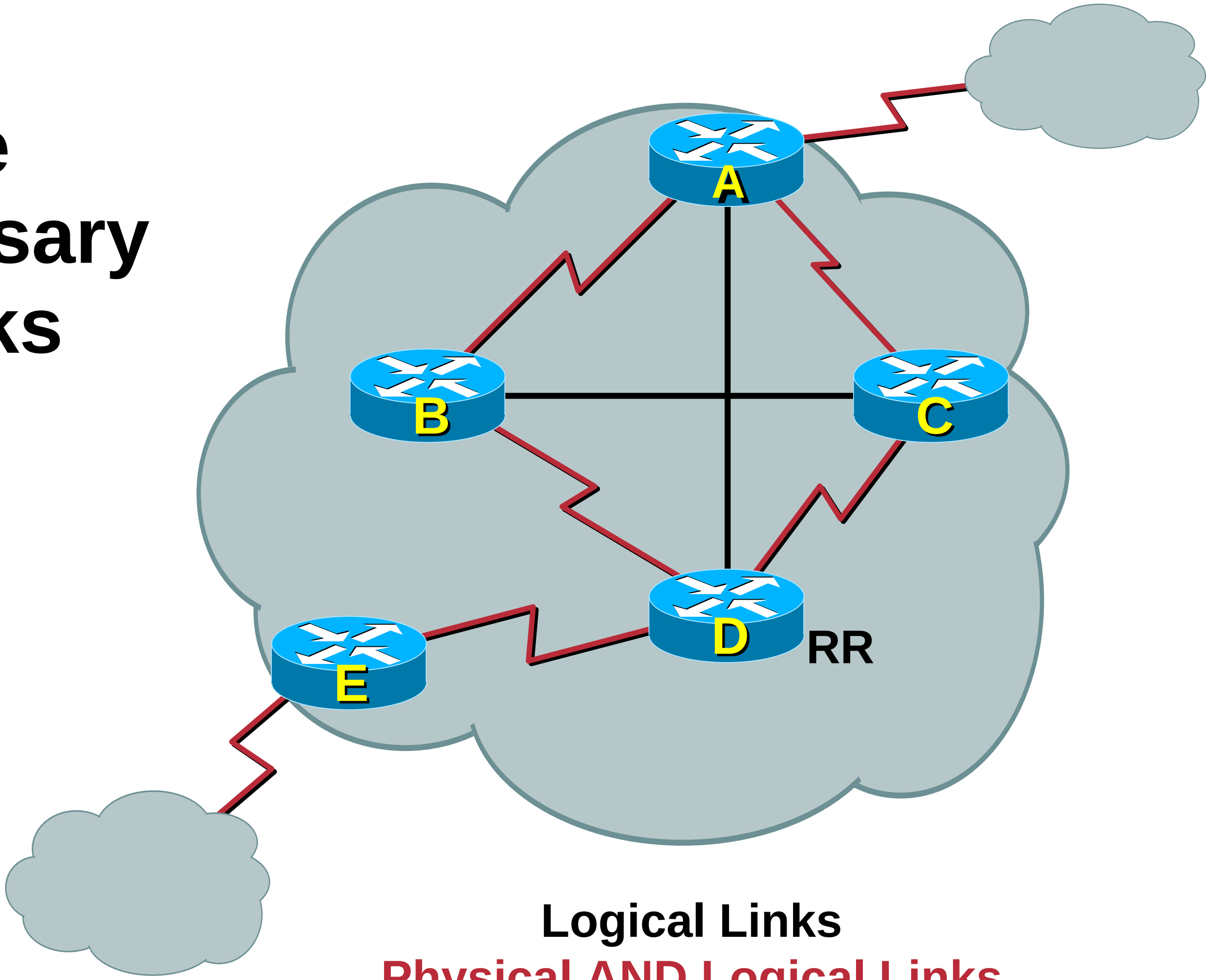


Logical Links

Physical AND Logical Links

Route Reflectors—Migration

- **Step 2: eliminate unnecessary iBGP links**

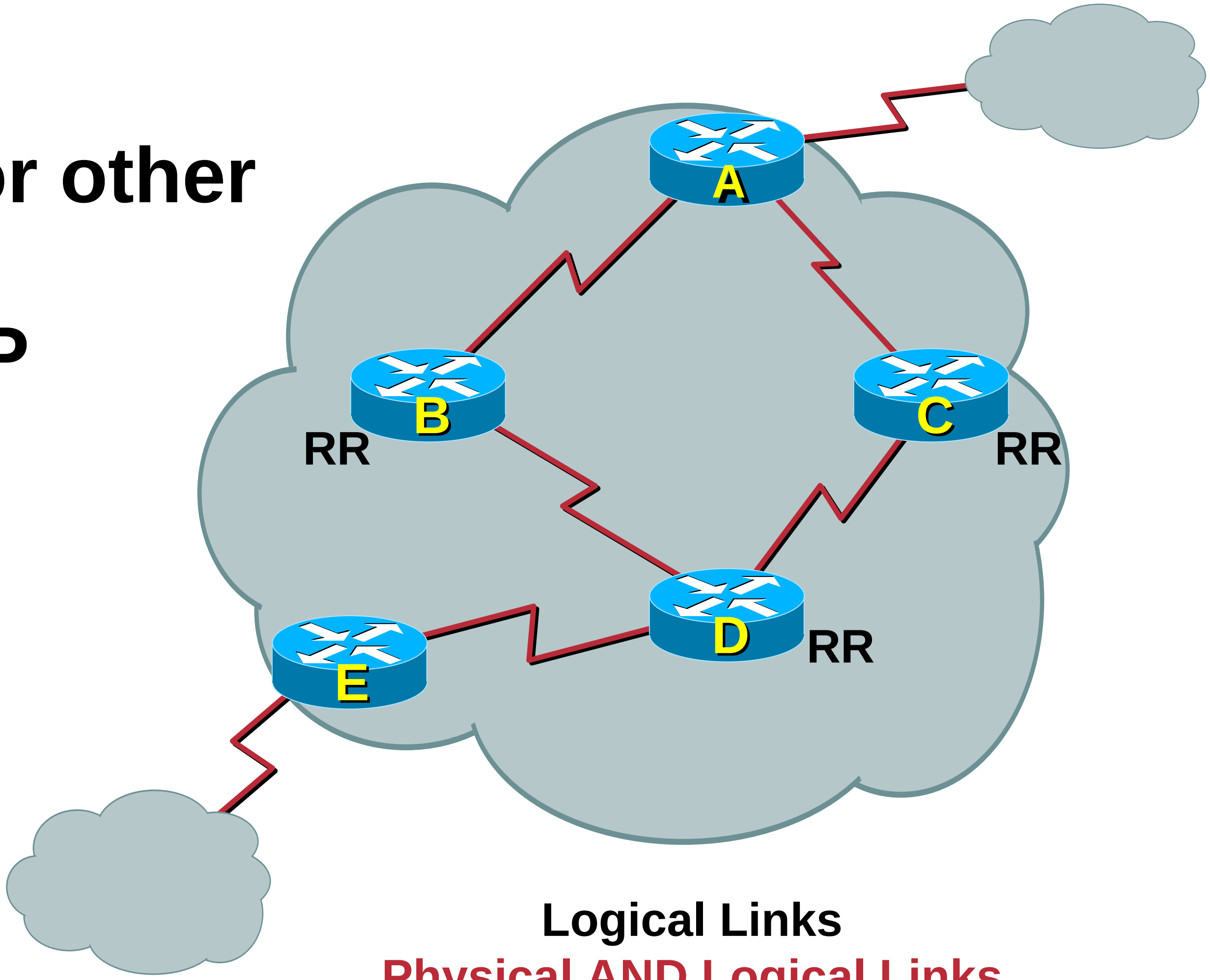


Logical Links

Physical AND Logical Links

Route Reflectors—Migration

- **Step 3: repeat for other clusters and iBGP links**



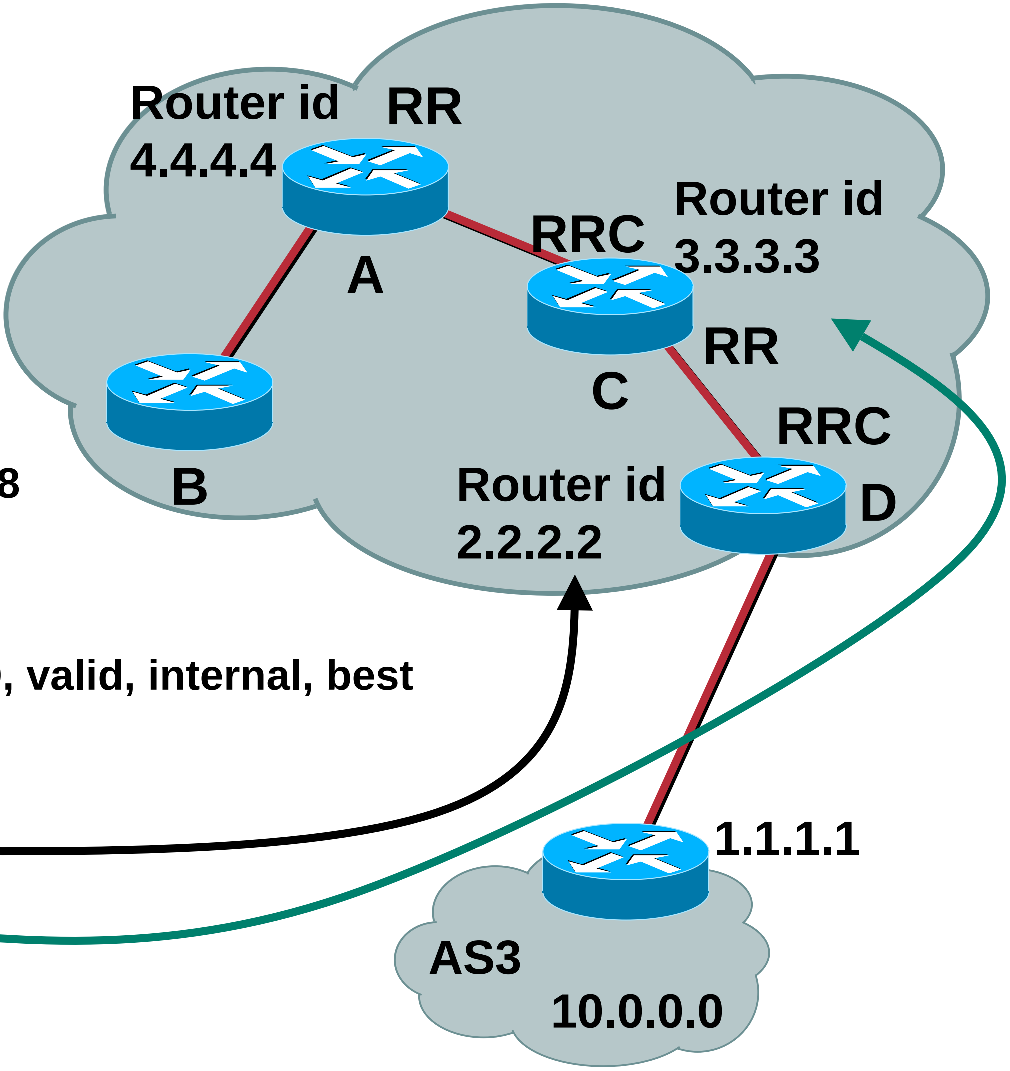
Logical Links

Physical AND Logical Links

Route Reflectors - Attributes

- **Example:**

```
RouterB>sh ip bgp 10.0.0.0
BGP routing table entry for 10.0.0.0/8
 3
   1.1.1.1 from 4.4.4.4 (2.2.2.2)
   Origin IGP, metric 0, localpref 100, valid, internal, best
```



Originator: 2.2.2.2

Cluster list: 4.4.4.4, 3.3.3.3

Route Reflectors - Redundancy

- A RRC may peer with more than one reflector, in different clusters.

  A RRC that peers to only one RR has a single point of failure

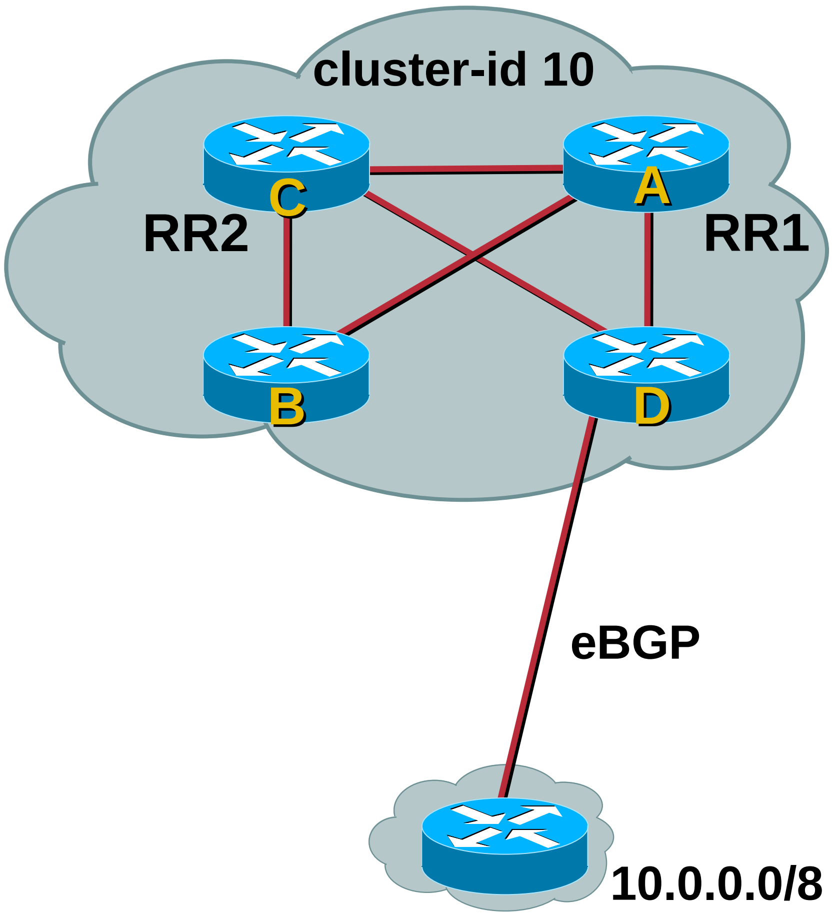
  RRC should peer to at least two RRs to provide redundancy

- The million dollar question

  Should redundant RRs be in the same cluster or should they be in separate clusters?

Same Cluster-ID

- RRs A and C have the same Cluster-ID
- C will deny routes reflected from A due to cluster-list loop detection
- If session from C to D fails, C will not be able to reach 10.0.0.0/8
- If session from B to A fails, B will not be able to reach 10.0.0.0/8
- D has some redundancy, but not 100%

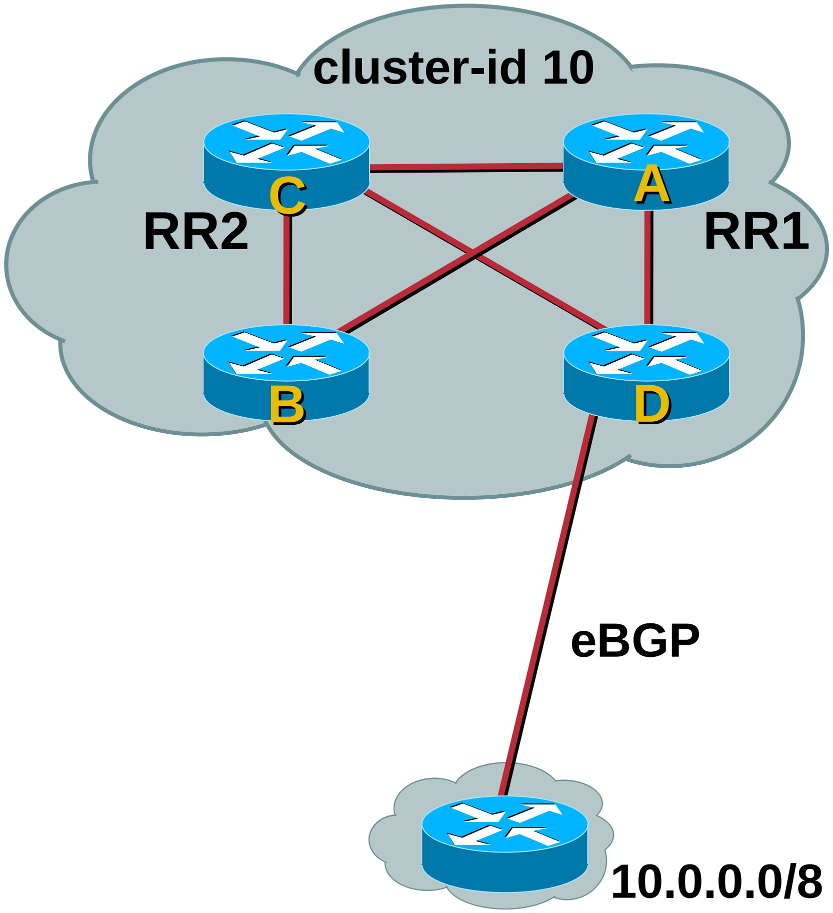


Lines Represent Both Physical Links and BGP Logical Connections

Same Cluster-ID

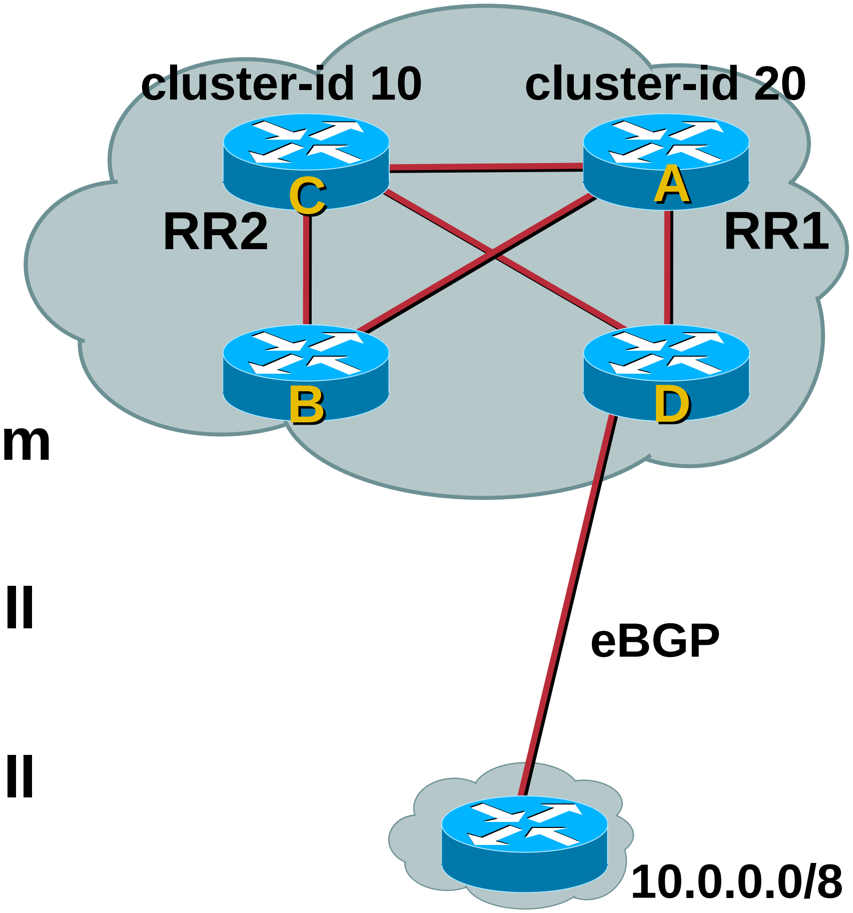
Cisco.com

- Technically not 100% redundant
- If loopback peering is used then the chances of C→D or B→A failure are greatly reduced
- Using same Cluster-ID with loopback peering should be ok



Different Cluster-ID

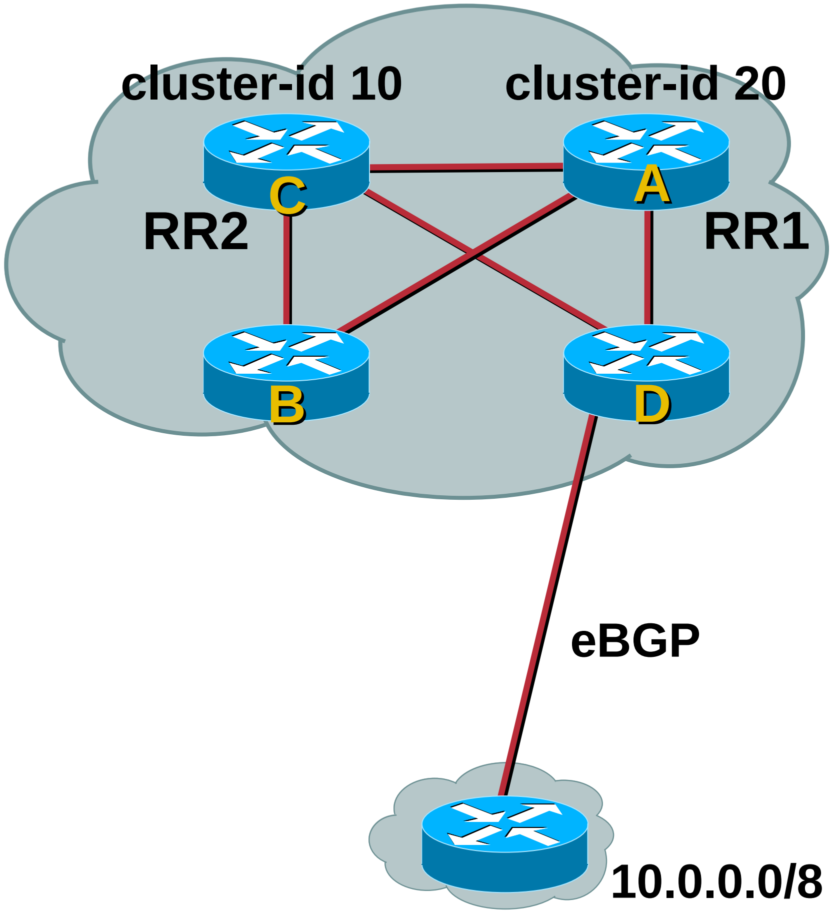
- RRs A and C have different Cluster-IDs
- C will not deny routes reflected from A
- C will know about 10.0.0.0/8 from A and D
- If C to D session fails, C can still reach 10.0.0.0/8 via A
- If B to A session fails, B can still reach 10.0.0.0/8 via C
- D has true redundancy



Lines Represent Both Physical Links and BGP Logical Connections

Different Cluster-ID

- C has two paths to 10.0.0.0/8 but only had one path in “Same Cluster-ID” topology
- Unique Cluster-IDs mean more paths on RRs which translates to more memory

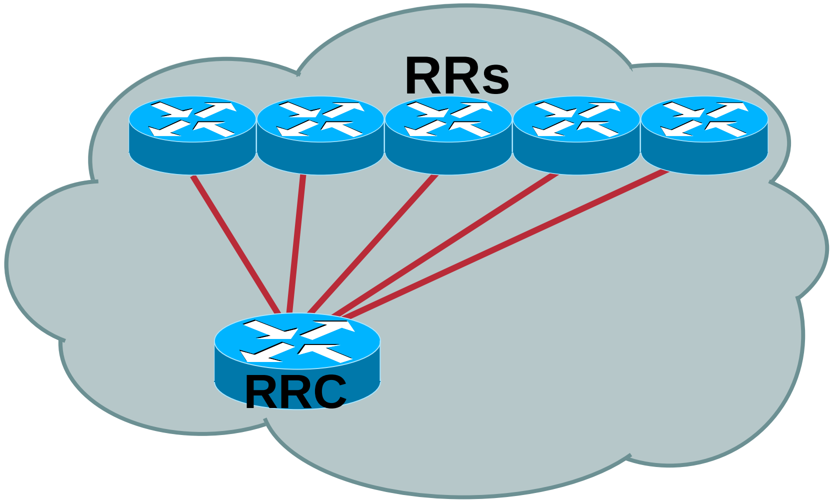


Cluster-ID Comparison

	Redundancy	Admin Factors		Attribute Combos	RR Memory Consumption
Same Cluster-ID	~100% with loopback peering	Easy to ID POPs based on Cluster-ID		Medium	One path from each RRC
Different Cluster-ID	100%	Easy to ID router based on Cluster-ID		High	One path from each RRC and one from each RR

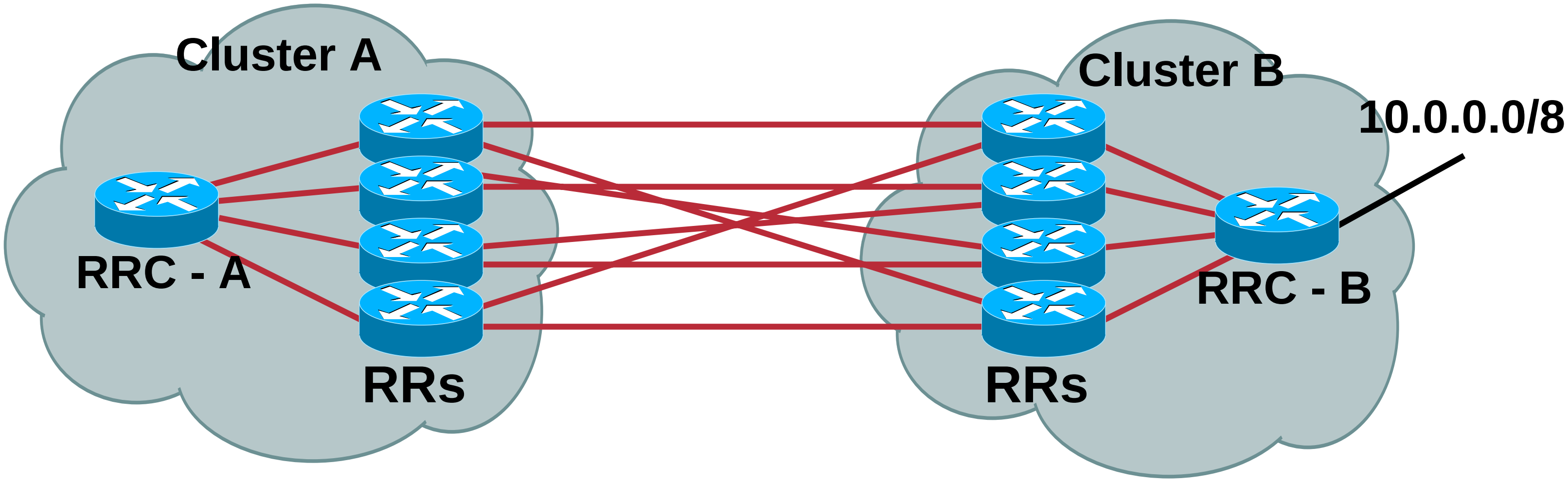
Route Reflectors - Redundancy

- Can a RRC have too much redundancy?
- RRC will receive an additional view for each extra RR it peers with, which will consume extra memory.



Route Reflectors - Redundancy

- Each RR in Cluster A has 4 paths to 10.0.0.0/8
- Only one exit point for this prefix but we learn about it from 4 peers
- Increases memory consumption on RRs

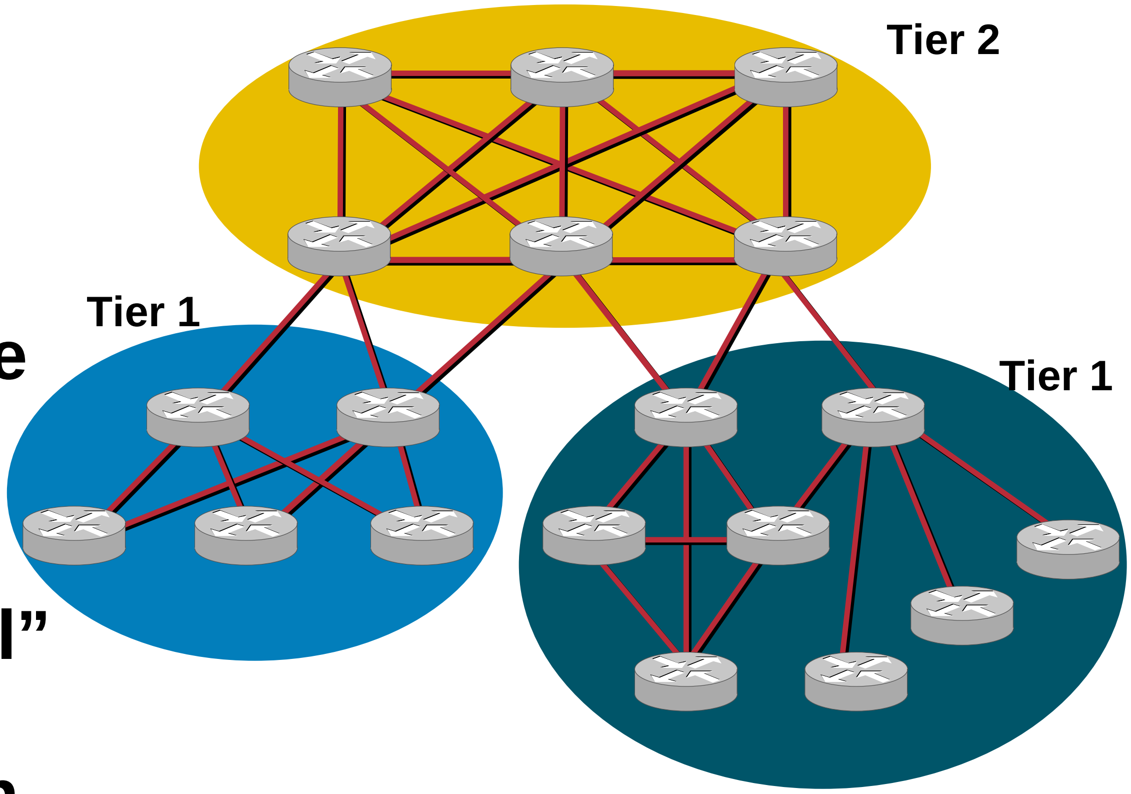


Route Reflectors - Redundancy

- Some redundancy is needed
- Too much burns memory on RRCs because the client learns the same information from each RR
- Also burns memory on the RRs because they learn multiple paths for each route introduced by a RRC
- Two or three reflectors peer cluster should be plenty

Route Reflectors - Hierarchy

- Clusters may be configured hierarchically
- RRs in a cluster are clients of RRs in a higher level
- Provides a “natural” method to limit routing information sent to lower levels



Protocol Issues - Agenda

iBGP Full Mesh

Route Reflectors

Confederations

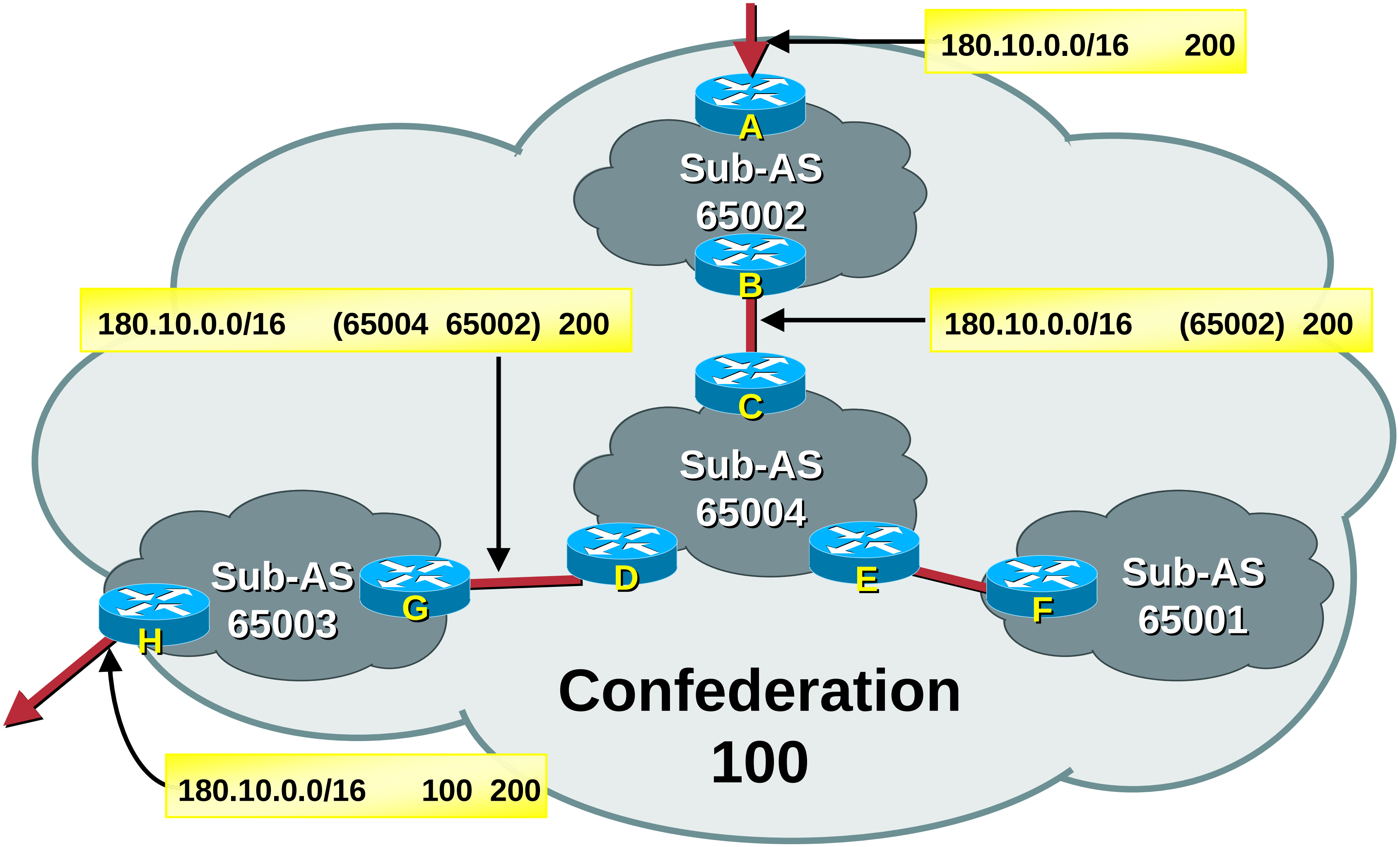
Detection and Propagation of Changes

- minRouteAdvertisementInterval
- NEXT_HOP Reachability
- Route Dampening

Confederations

- Described in RFC 3065.
- An AS is split into multiple Sub-ASes; still looks like a single AS to eBGP peers.
  - Sub-AS numbers should come from private AS range
- BGP sessions between each Sub-AS are similar to eBGP
  - Preserve NEXT_HOP, LOCAL_PREF and MED.
  - AS_CONFED_SEQUENCE is used to perform loop detection.

Confederations – AS_CONFED_SEQ



Confederations – Deployment

- No graceful way to migrate an existing network from a full mesh to confederations.
- Easy to define policies per Sub-AS.

  "independent sub-AS administration"

- NEXT_HOP can be reset when advertising routes from one Sub-AS to another

  Makes it possible to run a separate IGP per Sub-AS!!

- Provides quick and dirty method of integrating a network into an existing one.

Confederations – Summary

- ***Simplify*** **the network topology.**

  Allow contained hierarchy per sub-AS.

- **Policy may be defined per sub-AS**

  Ease of network integration.

- **Migration to/from confederations is not straight forward.**

RRs or Confederations

	External Connectivity	Multi-Level Hierarchy	Policy Control	Scalability	Migration Complexity
Confederations	Anywhere In the Network	Yes	Yes	Medium	Medium To High
Route Reflectors	Anywhere In the Network	Yes	Yes	Very High	Very Low

Protocol Issues - Agenda

iBGP Full Mesh

Route Reflectors

Confederations

Detection and Propagation of Changes

- minRouteAdvertisementInterval
- NEXT_HOP Reachability
- Route Dampening

minRouteAdvertisementInterval

> “MinRouteAdvertisementInterval determines the minimum amount of time that must elapse between advertisement of routes to a particular destination from a single BGP speaker.”

draft-ietf-idr-bgp4-13

Section 9.2.3.1

minRouteAdvertisementInterval

- *Studies have been made to study the effects of the minRouteAdvertisementInterval on BGP convergence
- In a nutshell

  Keeping the timer per peer instead of per prefix has some negative effects

  The default MinAdvInterval of 30 seconds may be too long

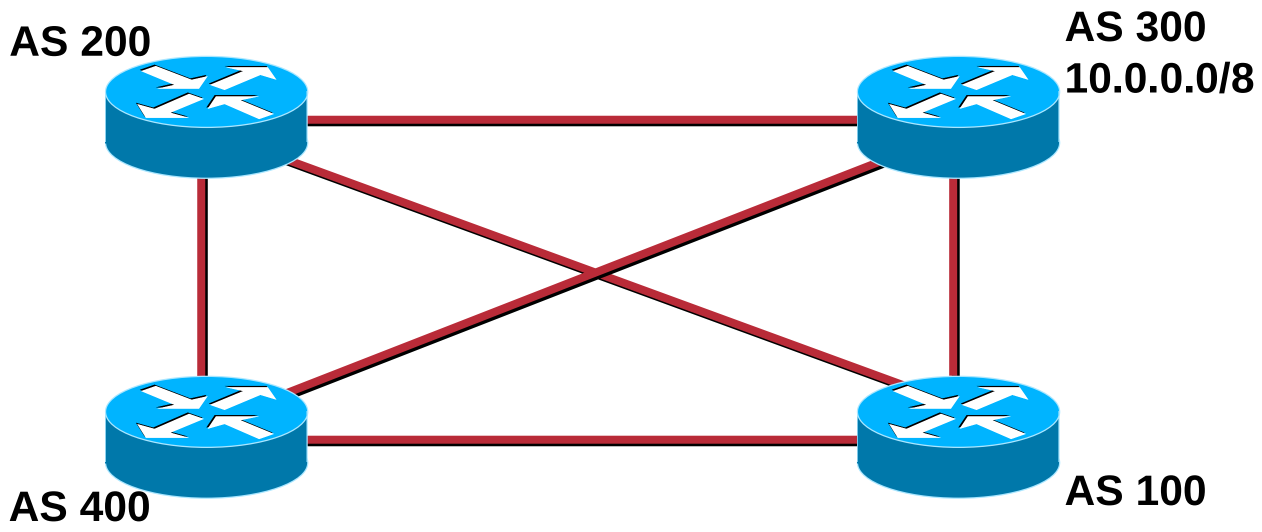
  TX Loop Detection should be implemented

    using an outbound filter to prevent advertising routes to a peer that will deny them due to AS_PATH loop detection

***** “An Experimental Study of Internet Routing Convergence”**

- Labovitz, Ahuja, Bose, Jahanian

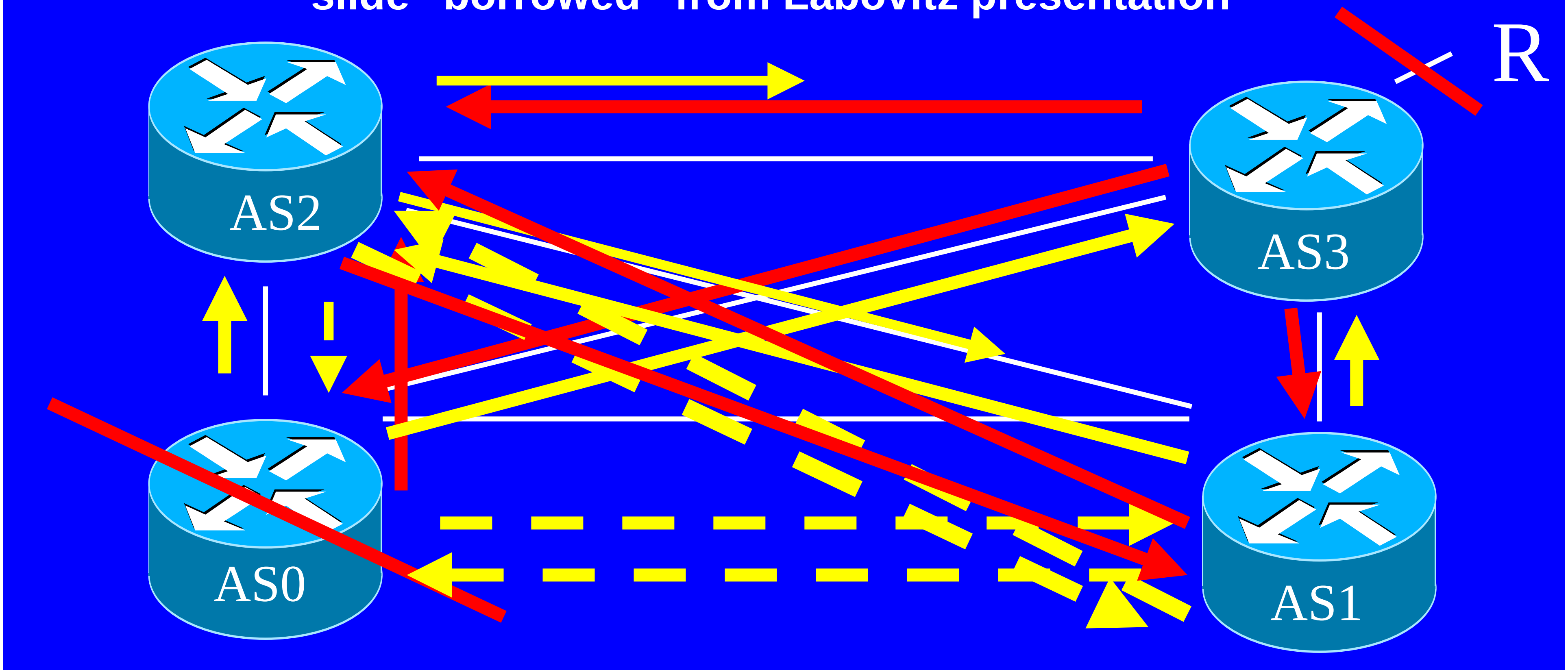
minRouteAdvertisementInterval



- Topology used to perform internal testing to study the effects when flapping the 10.0.0.0/8 prefix
- Convergence time, number of messages sent, number of denied messages, etc… are all monitored

BGP Convergence Example

- slide “borrowed” from Labovitz presentation



~~*B R via 3~~
~~*B R via 13~~
~~B R via 23~~

AS0

~~*B R via 3~~
~~*B R via 03~~
~~*B R via 203~~

AS1

~~*B R via 3~~
~~*B R via 013~~
~~B R via 103~~

AS2

Min Adv Interval - Variables

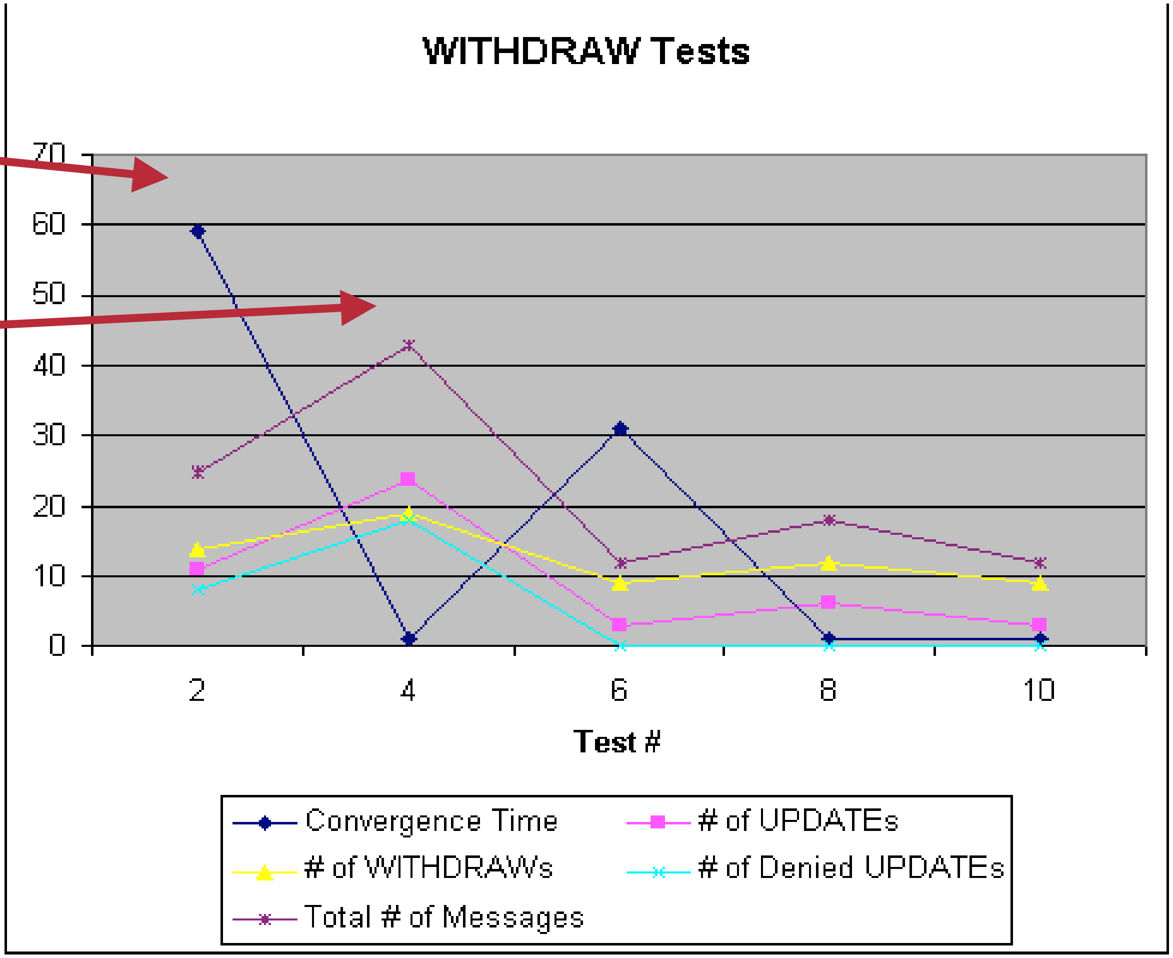
- Min Adv Interval – 0 seconds, 1 second, and 30 seconds
- Message Type – Advertisement (UPDATE) or WITHDRAW
- TX Loop Detection – Either On or Off. Refers to using an outbound filter to prevent advertising routes to a peer that will be denied due to AS_PATH loop detection. Example: If peer A is in AS 100 do not send A any routes that have AS 100 in the AS_PATH.

minRouteAdvertisementInterval -- Test Matrix

	Message Type.	Timer (sec)	TX Loop Detection	# Msgs Total	Denied UPDATES	Conv. (sec)
Test 1	UPDATE	30		9		< 1
Test 2	WITHDRAW	30		25	8	59
Test 3	UPDATE	0		9		< 1
Test 4	WITHDRAW	0		43	18	< 1
Test 5	UPDATE	30	X	9		< 1
Test 6	WITHDRAW	30	X	12		31
Test 7	UPDATE	0	X	9		< 1
Test 8	WITHDRAW	0	X	18		< 1
Test 9	UPDATE	1	X	9		< 1
Test 10	WITHDRAW	1	X	12		< 1

minRouteAdvertisementInterval - Conclusions

- Default behavior takes almost 1 minute to converge
- Using a MinAdvInterval of 0 results results in a flurry of messages (43) for a single route-flap (see Test 4)
- Using TX Loop Detection reduces the number of messages sent (see Tests 6, 8, and 10)
- Best results are in test 10 which uses TX Loop Detection with Min Adv Interval of 1 second



minRouteAdvertisementInterval - Conclusions

- Withdrawal messages are not subject to timer
- Sending UPDATEs that will be denied unnecessarily triggers timer
- Setting the timer to 0 causes a flurry of messages

NEXT_HOP Reachability

- **The NEXT_HOP MUST be reachable for the BGP path to be valid.**

  Reachability should be provided by the IGP.

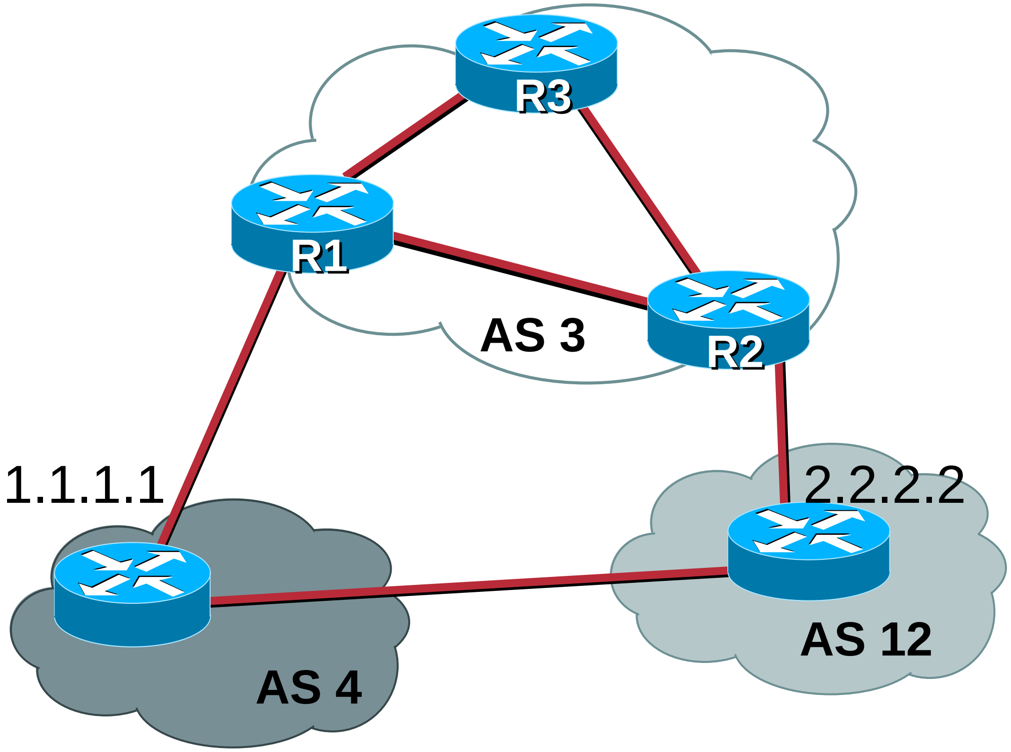
- **Other route characteristics also important for best path selection**

  IGP metric to NEXT_HOP

- **Change in the reachability characteristics of the NEXT_HOP (availability, cost) may impair the ability to forward traffic and/or cause black holes or routing loops.**

  BGP depends on the underlying IGP to provide fast and consistent notification of any change

NEXT_HOP Reachability



- R1 and R2 advertise routes to R3 with NEXT_HOPs of 1.1.1.1 and 2.2.2.2
- R3 must have a route to these two addresses
- Black Holes and severe route flapping can occur if R3 does not have a proper route to both NEXT_HOPs

NEXT_HOP Reachability

- **Four solutions**
- **Option 1 - Carry the R1 and R2 eBGP peering links in the IGP**

  Adds extra routes to the IGP

  Carrying customer links adds instability to the IGP

- **Option 2 - Do "redistribute connected" and "redistribute static" into BGP on R1 and R2**

  Adds a LOT of extra routes to BGP. Connected subnets of any router with an eBGP peer are now carried in the IGP and BGP

  Carrying customer links adds instability to BGP

  BGP will know how to get to its BGP NEXT_HOPs via BGP. Illegal recursive lookups can easily led to severe route churn

  Two recursive lookups have to be done to resolve the outbound interface. Traffic forwarding is not effected but troubleshooting multiple recursive lookups becomes complex

NEXT_HOP Reachability

- **Option 3 - Do "*neighbor x.x.x.x next-hop-self*" on the iBGP sessions from R1 and R2 to R3**

  Adds 0 routes to the IGP

  Adds 0 routes to BGP

  Promotes IGP/BGP stability by leaving customer links out of the picture

  BGP will have an IGP route to BGP NEXT_HOPs. Route churn due to illegal recursive lookups is no longer an issue

  NEXT_HOPs accessed via a single recursive lookup which makes troubleshooting easier

- **Note: "next-hop-self" to a route-reflector-client will not modify the NEXT_HOP of a reflected route. Routes advertised from an eBGP peer to a RRC will be modified**

NEXT_HOP Reachability

- **Option 4 – Do “redistribute static/connected” into BGP but overwrite next-hop to self for most prefixes**
- **Common practice to redistribute customer links into BGP and then tweak IGP metric as a means of moving traffic off an over-loaded peer**
- **Ideal solution is to leave the next-hop alone for these overcrowded peers but do next-hop-self for all other prefixes**
- **Is possible by applying outbound route-map on all iBGP peers**

NEXT_HOP Reachability

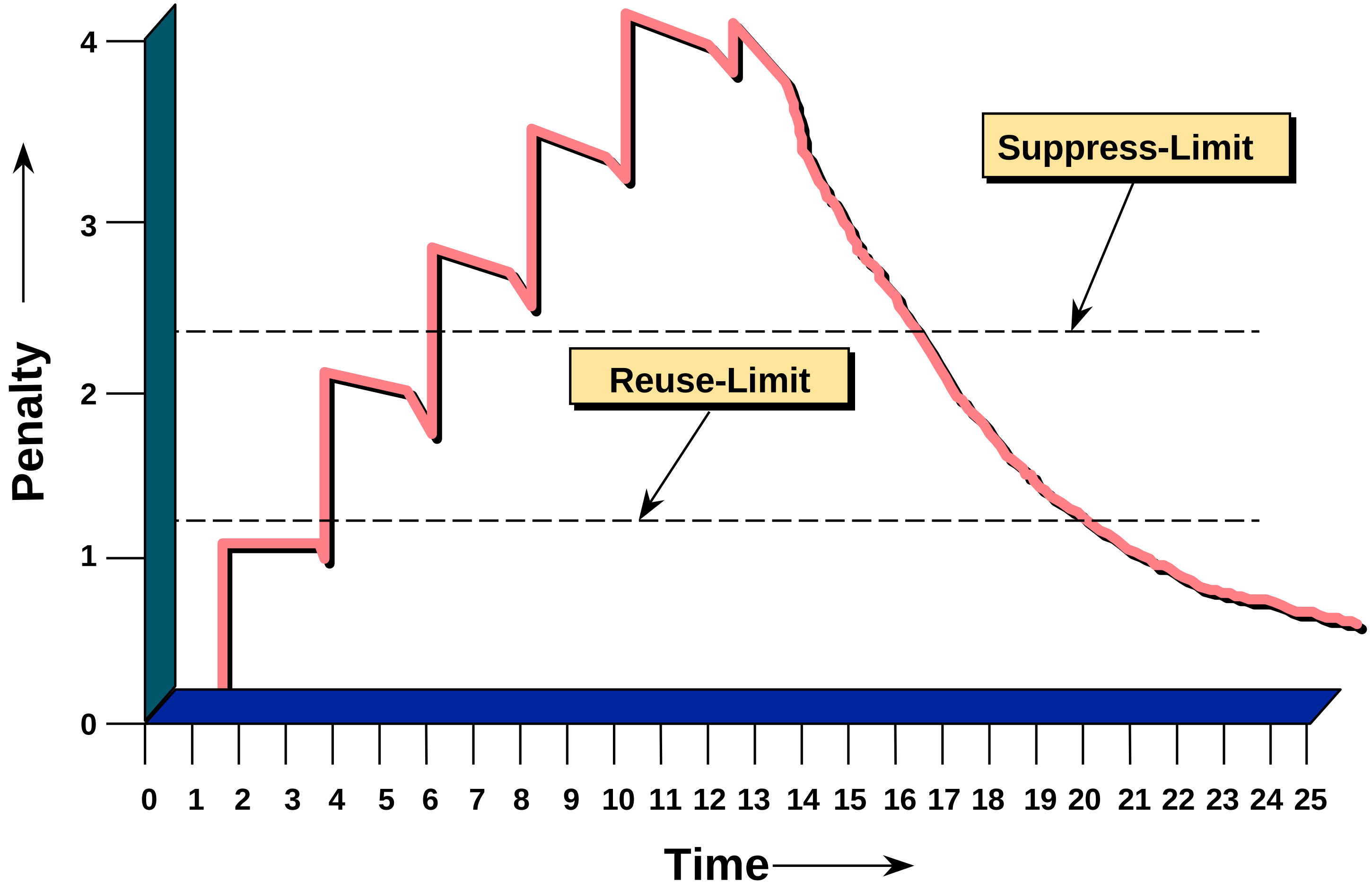
- router bgp 100
- neighbor iBGP route-map foo out
- neighbor iBGP-clients route-map foo out

- ip access-list standard CONGESTED_NEXTHOPS
- permit 10.1.1.3
- !
- ! Do not modify next-hop for congested peers
- route-map foo permit 10
- match ip next-hop CONGESTED_NEXTHOPS
- !
- ! Do next-hop-self on everything else, except the internal routes that are reflected
- route-map foo permit 20
- match route-type external
- set ip next-hop peer-address
- !
- ! Allow everything else to pass
- route-map foo permit 30

Dampening

- Defined in RFC 2439.
- Route flap: The bouncing up and down of a path or a change in its characteristics.
  - A flap ripples through the entire Internet
  - Consumes CPU cycles, causes instability
- Solution: Reduce scope of route flap propagation
  - History predicts future behavior
  - Suppress oscillating routes
  - Advertise stable suppressed routes
  - Only external routes are dampened.

Dampening


Penalty
4
3
2
1
0
Suppress-Limit
Reuse-Limit
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Time

Dampening

- **A route can only be suppressed when receiving an advertisement.**

  Not when receiving a WITHDRAW.

  Attribute changes count as a flap (1/2).

- **In order for a route to be suppressed the following must be true:**

  The penalty must be greater than the suppress-limit

  An advertisement for the route must be received while the penalty is greater than the suppress-limit

  A route will not automatically be suppressed if the suppress-limit is 1000 and the penalty reaches 1200. The route will only be suppressed if an advertisement is received while the penalty is decaying from 1200 down to 1000.

Dampening – Deployment

- Configurable parameters:

  half-life – The number of minutes it takes for the penalty to decay by 1/2

  reuse-limit – If a route is suppressed the penalty must decay to this value to be unsuppressed

  suppress-limit – The penalty must be greater than this threshold when an advertisement is received for a route to be suppressed

  max-suppress-time – The maximum number of minutes a route may be suppressed

Dampening – Deployment

- **Calculated parameters:**

  max-penalty – The maximum penalty a route may have that will allow the penalty to decay to reuse-limit within max-suppress-time

  max-penalty = reuse-limit * 2^(max-suppress-time/half-life)

  If half-life is 30, reuse-limit is 800, and max-suppress-time is 60 then the max-penalty would be 3200. If we allowed the penalty to reach 3201 it would be impossible for the penalty to decay to 800 within 60 minutes.

IOS will generate a warning message if the max-penalty is above 20,000 or less than the suppress-limit.

Dampening – Example

- **Small suppress window:**

  Half-life of 30 minutes, reuse-limit of 800, suppress-limit of 3000, and max-suppress-time of 60

  max-penalty is 3200

- **Advertisement must be received while penalty is decaying from 3200 down to 3000 for the route to be suppressed**

  A 3 min 45 second (rough numbers) window exist for an advertisement to be received while decaying from 3200 to 3000.

Dampening – Example II

- **No window:**

  Half-life of 30 minutes, reuse-limit of 750, suppress-limit of 3000, and max-suppress-time of 60

  max-penalty = 750 * 2^(60/30) = 3000

  Here the max-penalty is equal to the suppress-limit

- **The penalty can only go as high as 3000.**

  The decay begins immediately, so the penalty will be lower than 3000 by the time an advertisement is received.

  A route could consistently flap several times a minute and never be suppressed

Agenda

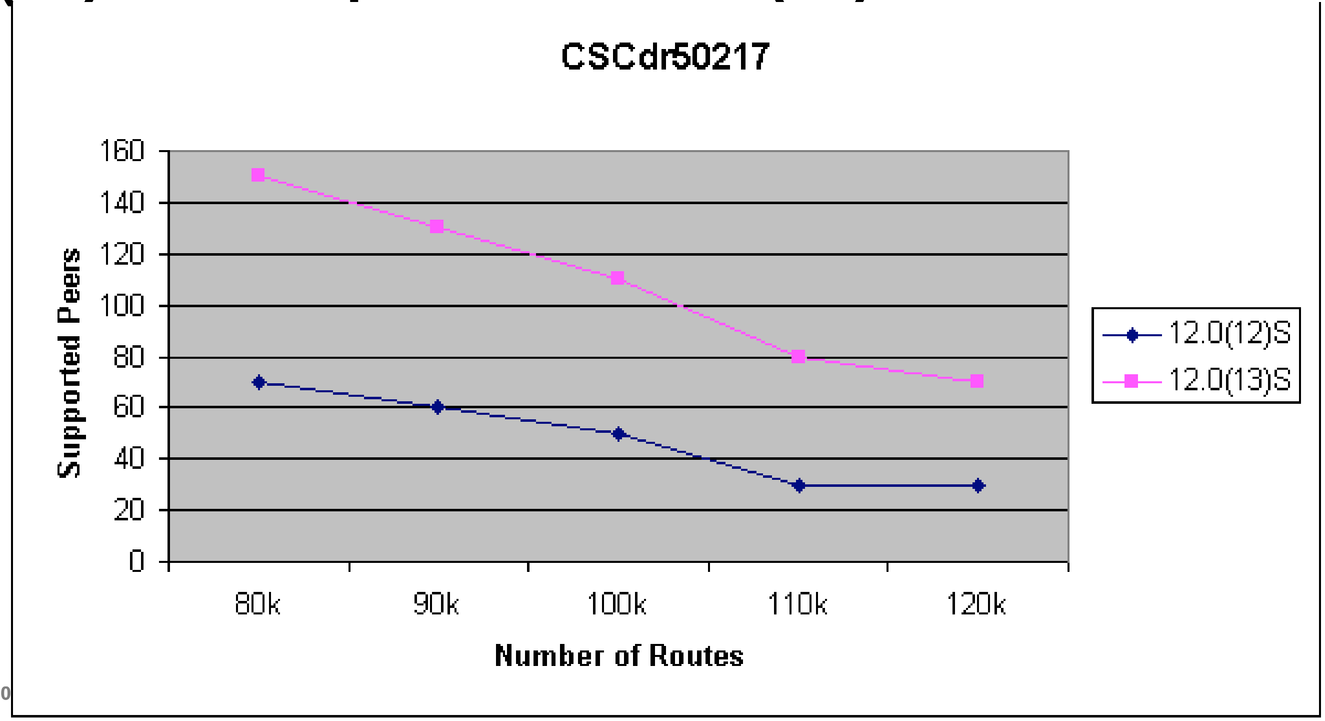
- **Scalability**
  - **Protocol Issues**
  - **Initial Convergence**
- **Troubleshooting**
  - **Peer establishment**
  - **Missing Routes**
  - **Inconsistent Route Selection**
  - **Loops and Convergence Issues**

Initial Convergence - Introduction

- Involves advertising 100k+ routes to hundreds of peers
- A vendor’s implementation of BGP plays a major roll in how fast a router can converge initially
- Will discuss bottlenecks/issues that we have found in the Cisco implementation

Before we begin…

- What does this graph show?
- Shows the number of peers we can converge in 10 minutes (y-axis) given a certain number of routes (x-axis) to advertise to those peers
- Example: We can advertise 100k routes to 50 peers with 12.0(12)S or 110 peers with 12.0(13)S



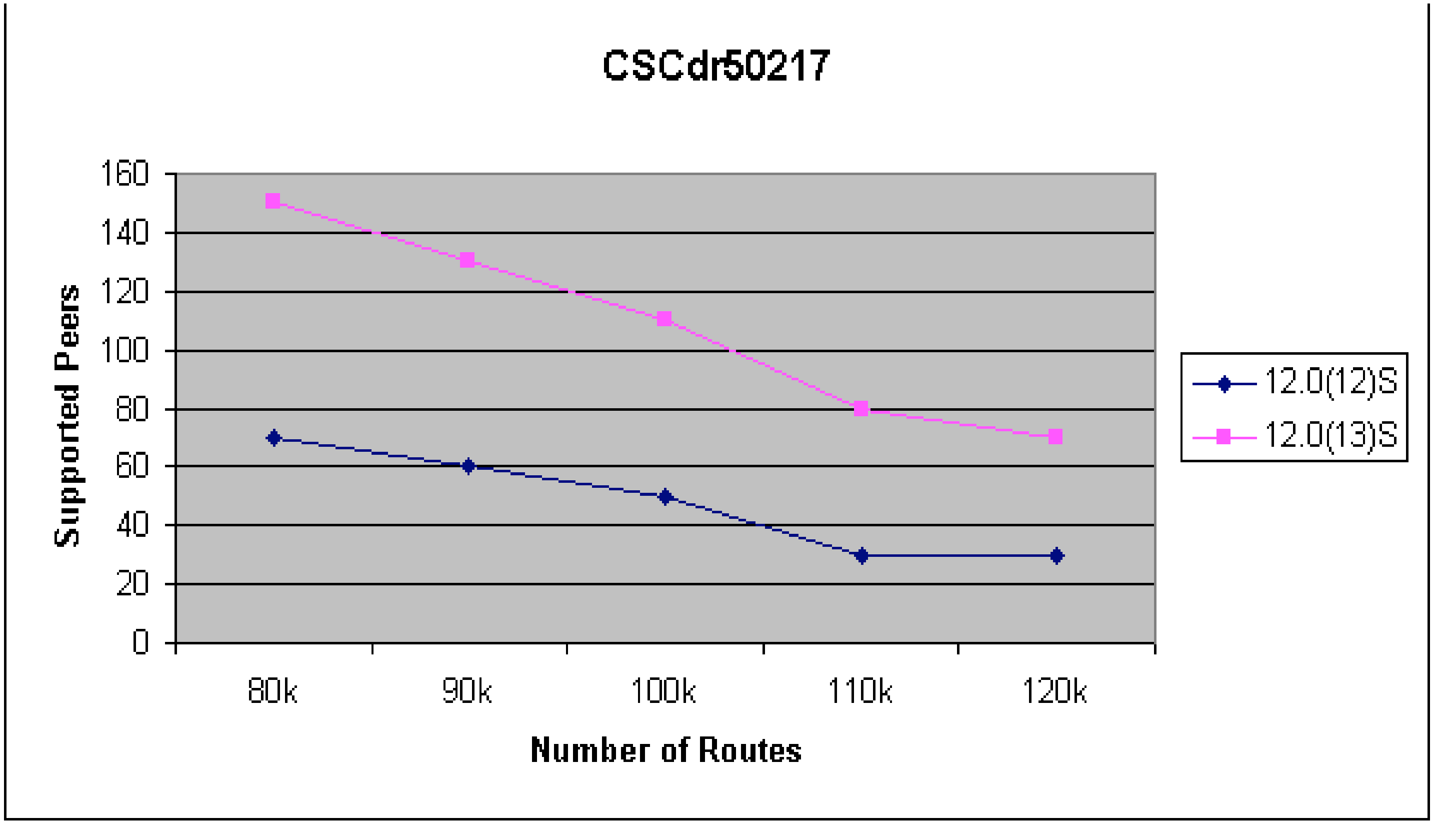
BGP → TCP Interaction

- CSCdr50217 – “BGP: Sending updates slow”
- Fixed in 12.0(13)S
- Description

  Fixed a problem in bgp_io which allows BGP to send data to TCP more aggressively

BGP → TCP Interaction

- **What does CSCdr50217 mean in terms of scalability?**
- **Almost 100% improvement**



Peer-groups

- Peer-groups address two scalability issues
  - Configuration size
  - UPDATE replication/advertisement
- A "peer-group" is a configuration tool that is used to apply the same commands to multiple peers without explicitly configuring those commands for each peer.
- Members of a peer-group will receive the same BGP UPDATEs. As a result, all members of a peer-group must have the same outbound policy.

Peer-groups

Configuration Example

```
neighbor 1.1.1.1 remote-as 100

neighbor 1.1.1.1 update-source
   Loopback 0

neighbor 1.1.1.1 send-community

neighbor 1.1.1.1 version 4

neighbor 1.1.1.2 remote-as 100

neighbor 1.1.1.2 update-source
   Loopback 0

neighbor 1.1.1.2 send-community

neighbor 1.1.1.2 version 4
```

```
! Define the peer-group

neighbor iBGP peer-group

neighbor iBGP remote-as 100

neighbor iBGP update-source
   Loopback 0

neighbor iBGP send-community

neighbor iBGP version 4

! Assign peers to the peer-group

neighbor 1.1.1.1 peer-group iBGP

neighbor 1.1.1.2 peer-group iBGP
```

BEFORE → AFTER

Peer-groups

Application Rules

- **All members MUST share a common outbound policy.**

  The same UPDATE message is sent to all the peers.

- **Examples:**

  RR-clients, but not a mixture of clients and iBGP peers

  iBGP OR eBGP peers, but not both in the same peer-group

  NEXT_HOP is an exception to the rule. Peers A and B can be in a peer-group and receive a different NEXT_HOP for an UPDATE. Accomplished by doing the *NEXT_HOP re-write* at the last minute

Peer-groups

- Three common eBGP peer-groups
  - Advertise default-route only
  - Advertise customer routes
  - Advertise full routes
- All should filter bogus inbound information
  - Address space that you use in your IGP!!
  - RFC 1918 address space
  - Class D and E addresses
  - Prefixes that are too specific (Class A /32s for example)
  - Un-assigned Class A, B, and C address space (optional)
  - “max-prefix” can be used for additional protection

Peer-groups

```
neighbor eBGP-default peer-group
neighbor eBGP-default route-map bogus_filter in
neighbor eBGP-default route-map default_only out
neighbor eBGP-default version 4
!
neighbor eBGP-customer peer-group
neighbor eBGP-customer route-map bogus_filter in
neighbor eBGP-customer route-map customer_routes out
neighbor eBGP-customer version 4
!
neighbor eBGP-full peer-group
neighbor eBGP-full route-map bogus_filter in
neighbor eBGP-full route-map full_routes out
neighbor eBGP-full version 4
```

Peer-groups

- **Problem: Advertise 100,000+ routes to hundreds of peers. BGP will need to send a few hundred megs of data in order to converge all peers.**
- **Solution: Use peer-groups!**

  UPDATE generation is done once per peer-group.

  The UPDATEs are then replicated for all peer-group member.

- ***Scalability is enhanced because more peers can be supported!***

Peer-groups

- UPDATE generation without peer-groups

  The BGP table is walked once, prefixes are filtered through outbound policies, UPDATEs are generated and sent…per peer!!

- UPDATE generation with peer-groups

  A peer-group *leader* is elected for each peer-group. The BGP table is walked once (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.

  Replicating an UPDATE is much easier/faster than formatting an UPDATE. Formatting requires a table walk and policy evaluation, replication does not.

Peer-groups

Synchronization

- A peer-group member is *synchronized* with the leader if all UPDATEs sent to the leader have also been sent to the peer-group member

  The more peer-group members stay in sync the more UPDATEs BGP can replicate.

- A peer-group member can fall out of sync for several reasons

  Slow TCP throughput

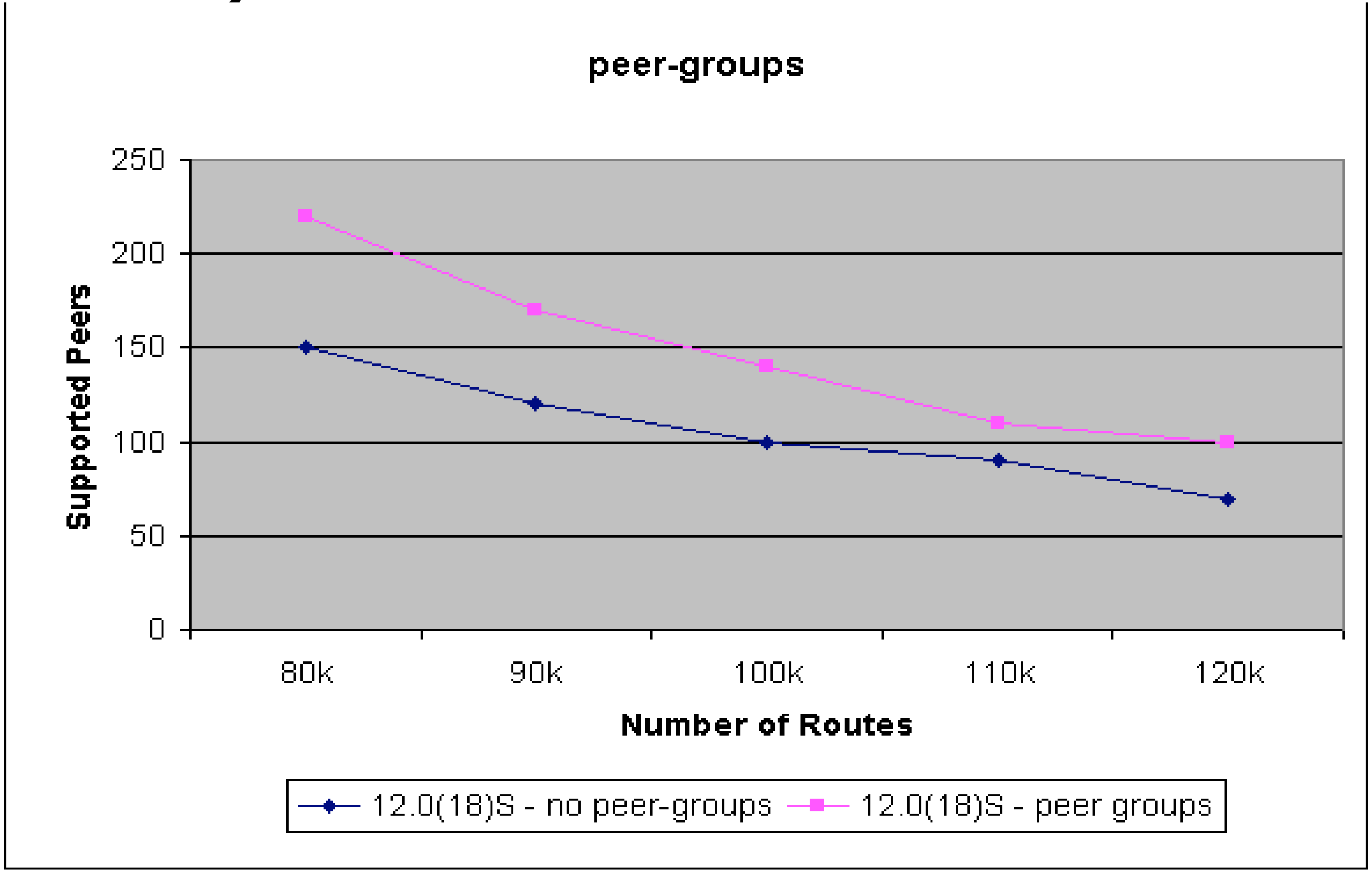
  Rush of TCP Acks fill input queues resulting in drops

  Peer is busy doing other tasks

  Peer has a slower CPU than the peer-group leader

Peer-groups

- **Peer-groups give between 35% - 50% increase in scalability**



Peer-groups – Summary

- **Peer-group scalability benefits:**

  UPDATE Generation and Replication

  Configuration Grouping

Larger Input Queues

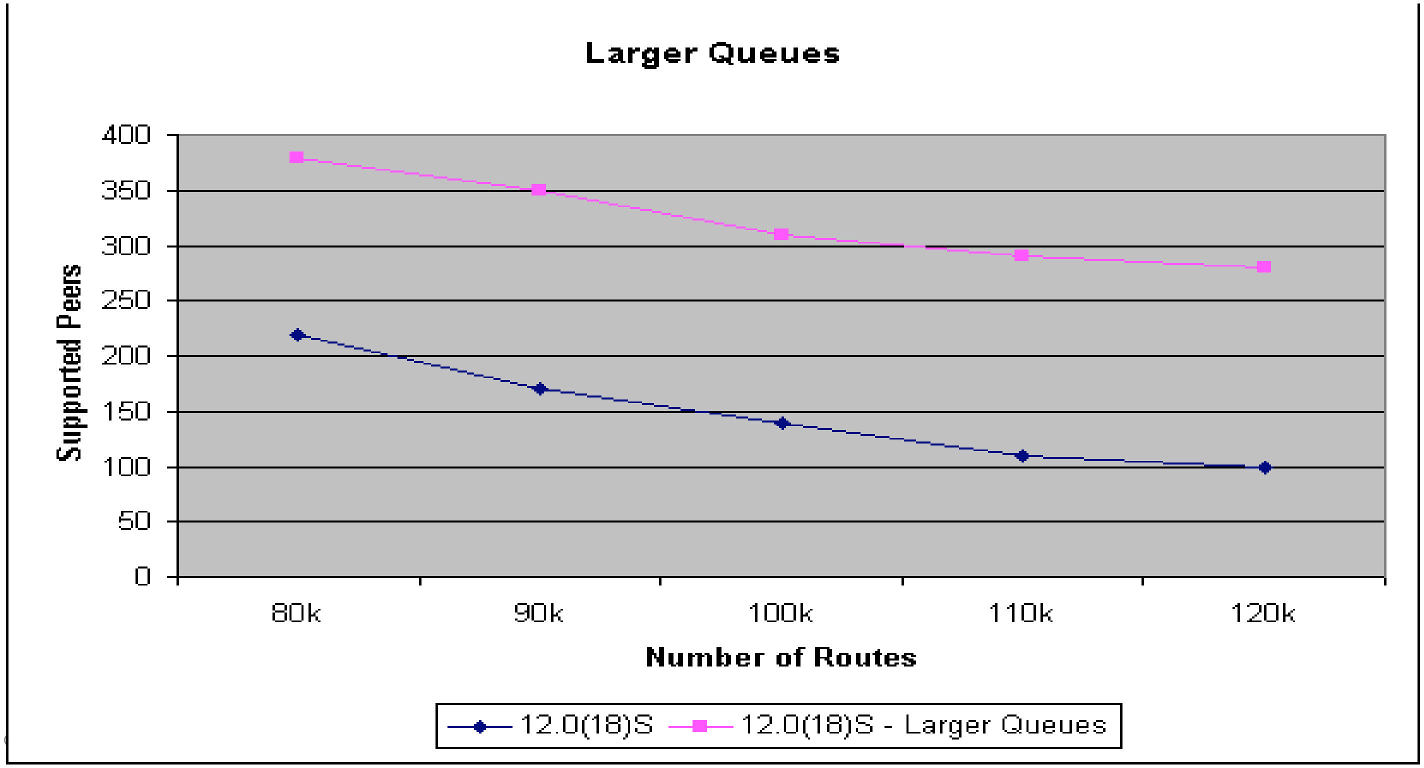
- In a nutshell

  If a BGP speaker is pushing a full Internet table to a large number of peers, convergence is degraded due to enormous numbers of drops (100k+) on the interface input queue. ISP foo gets ~½ million drops in 15 minutes on their typical route reflector.

- Increasing the size of the input queue, thus reducing the number of dropped TCP Acks, improves BGP scalability

Larger Input Queues

- Rush of TCP Acks from peers can quickly fill the 75 spots in process level input queues
- Increasing queue depths (4096) improves BGP scalability



Larger Input Queues

- **Why not change default input queue size?**

  May happen someday but people are nervous

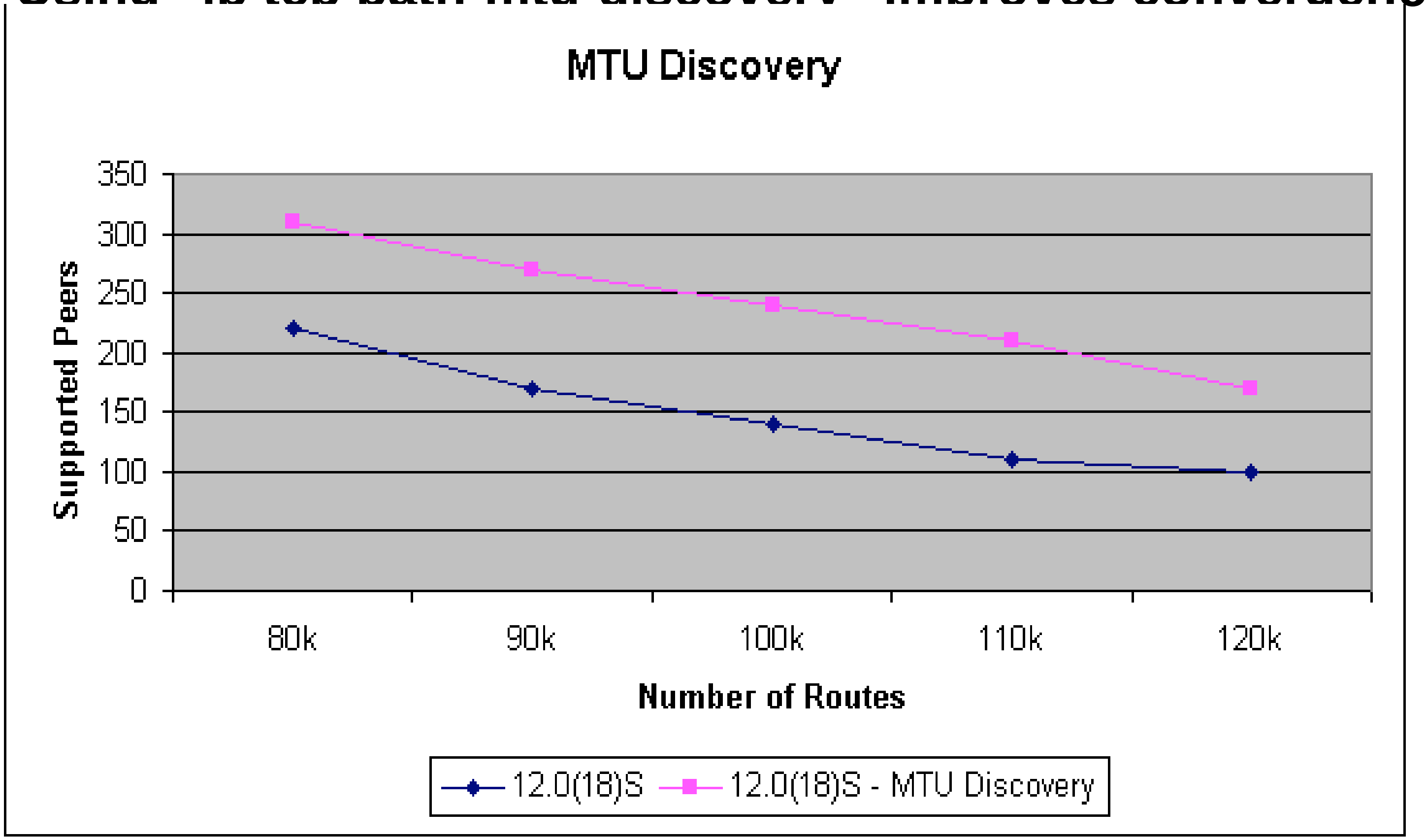
  CSCdu69558 has been filed for this issue

- **Even with 4096 spots in the input queue we can still see drops given enough routes/peers**

- **Need to determine "How big is too big" in terms of how large an input queue can be before we are processing the same data multiple times**

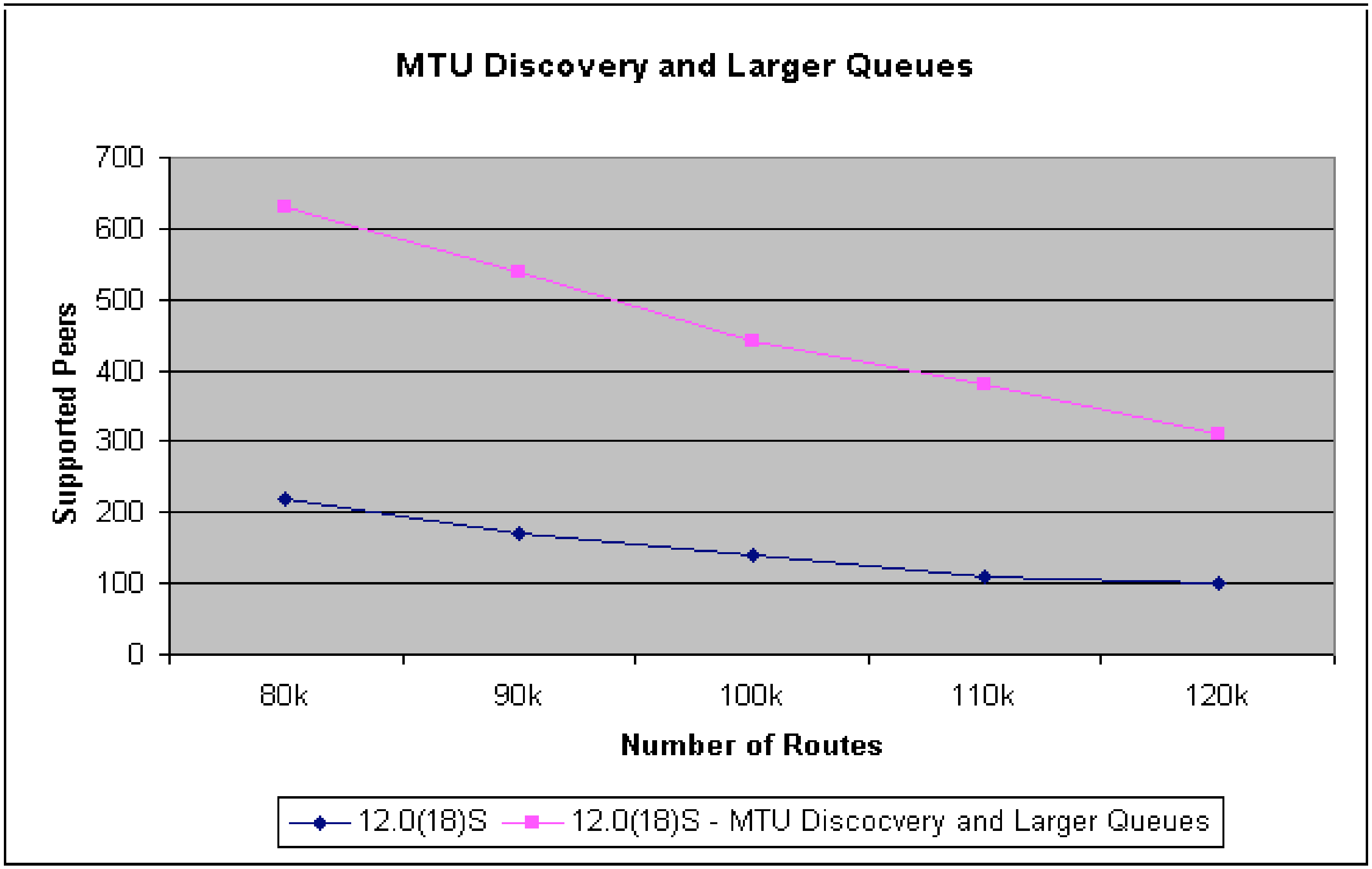
MTU Discovery

- Default MSS (Max Segment Size) is 536 bytes
- Inefficient for today’s POS/Ethernet networks
- Using “ip tcp path-mtu-discovery” improves convergence



MTU Discovery and Larger Input Queues

- **Simple config changes can give 3x improvement**



UPDATE Packing

- **Quick review on BGP UPDATEs**
- **An UPDATE contains:**

```
        +-----------------------------------------------------+
        |   Withdrawn Routes Length (2 octets)                |
        +-----------------------------------------------------+
        |   Withdrawn Routes (variable)                       |
        +-----------------------------------------------------+
        |   Total Path Attribute Length (2 octets)            |
        +-----------------------------------------------------+
        |   Path Attributes (variable)                        |
        +-----------------------------------------------------+
        |   Network Layer Reachability Information (variable) |
        +-----------------------------------------------------+
```

- **At the top you list a combination of attributes (MED = 50, Local Pref = 200, etc)**
- **Then you list all of the NLRI (prefixes) that share this combination of attributes**

UPDATE Packing

- If your BGP tables contains 100k routes and 15k attribute combinations then you can advertise all the routes with 15k updates if you pack the prefixes 100%
- If it takes you 100k updates then you are achieving 0% update packing
- Convergence times vary greatly depending on the # of attribute combinations used in the table and on how well BGP packs updates
- Ideal Table

  Routem generated BGP table of 75k routes

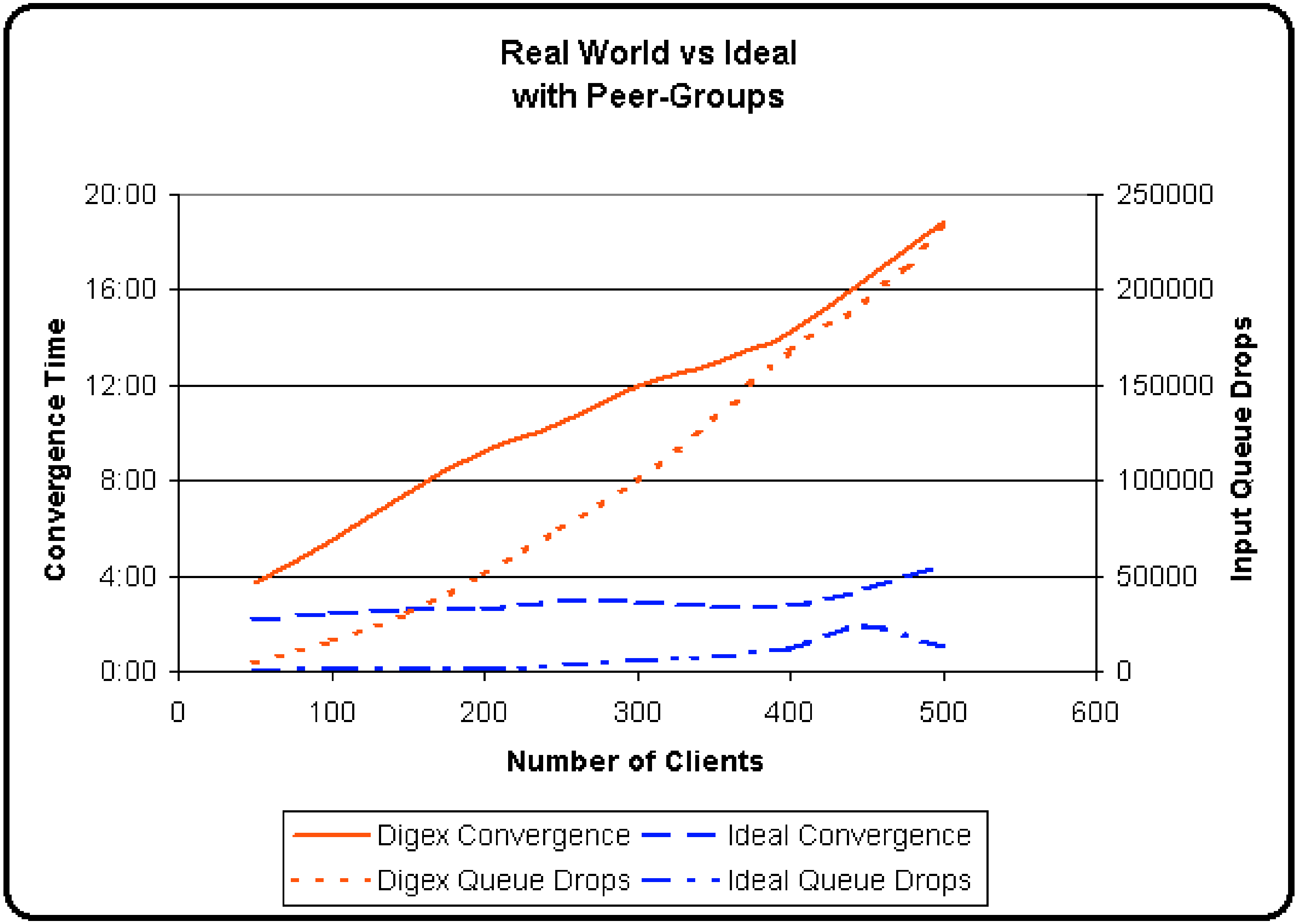
  All paths have the same attribute combination
- Real Table

  75k route feed from Digex (replayed via routem)

  ~12,000 different attribute combinations

UPDATE Packing

Cisco.com



UPDATE Packing

- With the ideal table we are able to pack the maximum number of prefixes into each update because all prefixes share a common set of attributes.
- With the real world table we send updates that are not fully packed because we walk the table based on prefix but prefixes that are side by side may have different attributes. We can only walk the table for a finite amount of time before we have to release the CPU so we may not find all the NLRI for a give attribute combination before sending the updates we have built and suspending.
- With 500 RRCs the ideal table takes ~4 minutes to converge where a real world table takes ~19 minutes!!

UPDATE Packing

- **UPDATE packing bugs**
- **BGP would pack one NLRI per update unless “set metric” was configured in an outbound route-map**

  CSCdt81280 - BGP: Misc fixes for update-generation – 12.0(16.6)S

  CSCdv52271 - BGP update packing suffers with confederation peers – 12.0(19.5)S

- **Same fix but CSCdt81280 is for regular iBGP and CSCdv52271 is for confed peers**

UPDATE Packing

- Example of CSCdt81280 from customer router
- BGP has 132k routes and 26k attribute combinations
- Took 130k messages to advertise 132k routes

```
•  132853 network entries and 1030454 paths using 49451673 bytes of memory
•  26184 BGP path attribute entries using 1361568 bytes of memory
•
•  Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
•  1.1.1.1         4   100      19  130681   354811    0    0 00:20:31       34
•  1.1.1.2         4   100     816  130782   354811    0    0 00:21:04     2676
```

UPDATE Packing

Cisco.com

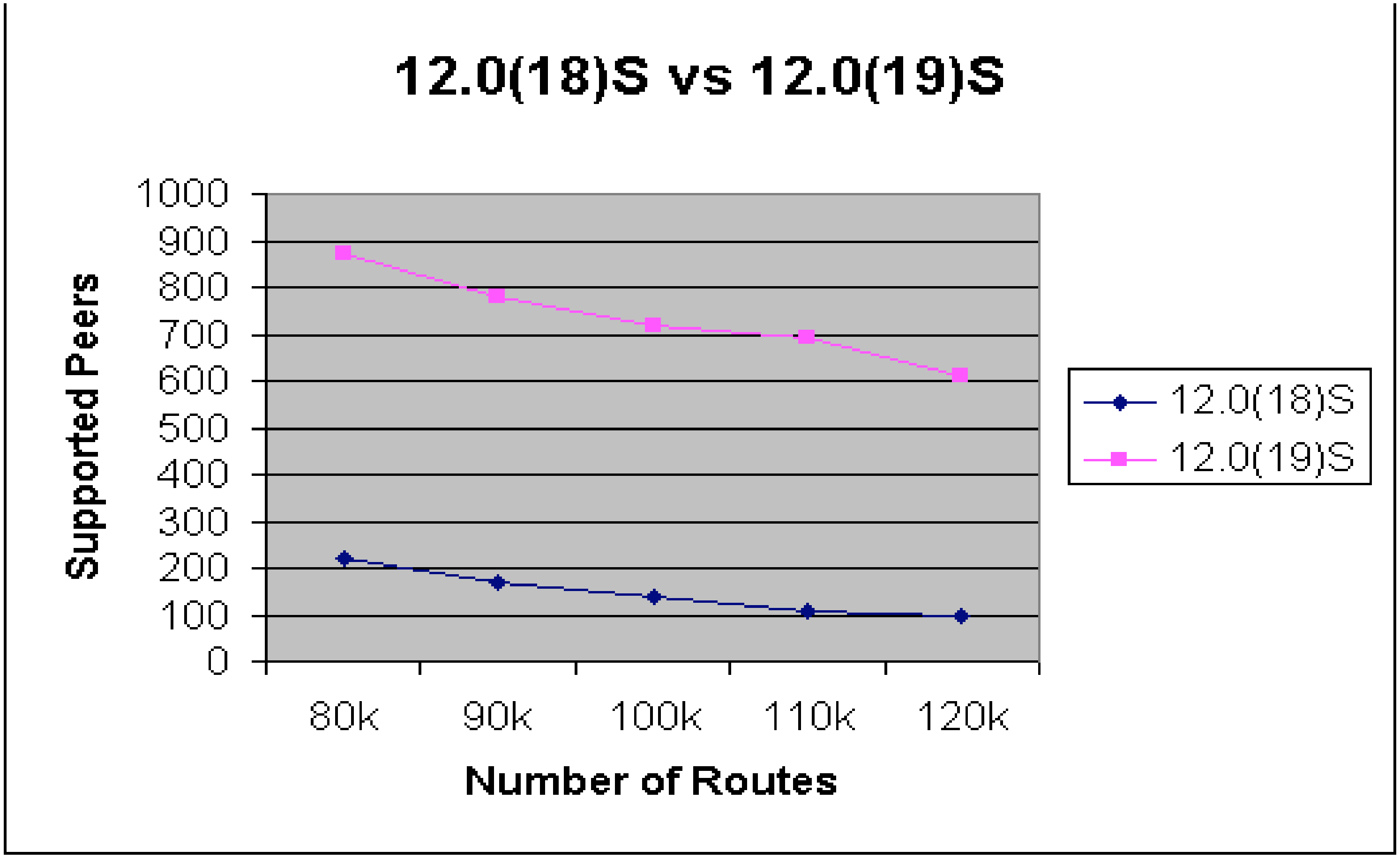
- CSCdt34187 introduces an improved update generation algorithm:

  100% update packing – attribute distribution no longer makes a significant impact

  100% peer-group replication – no longer have to worry about peers staying "in sync"

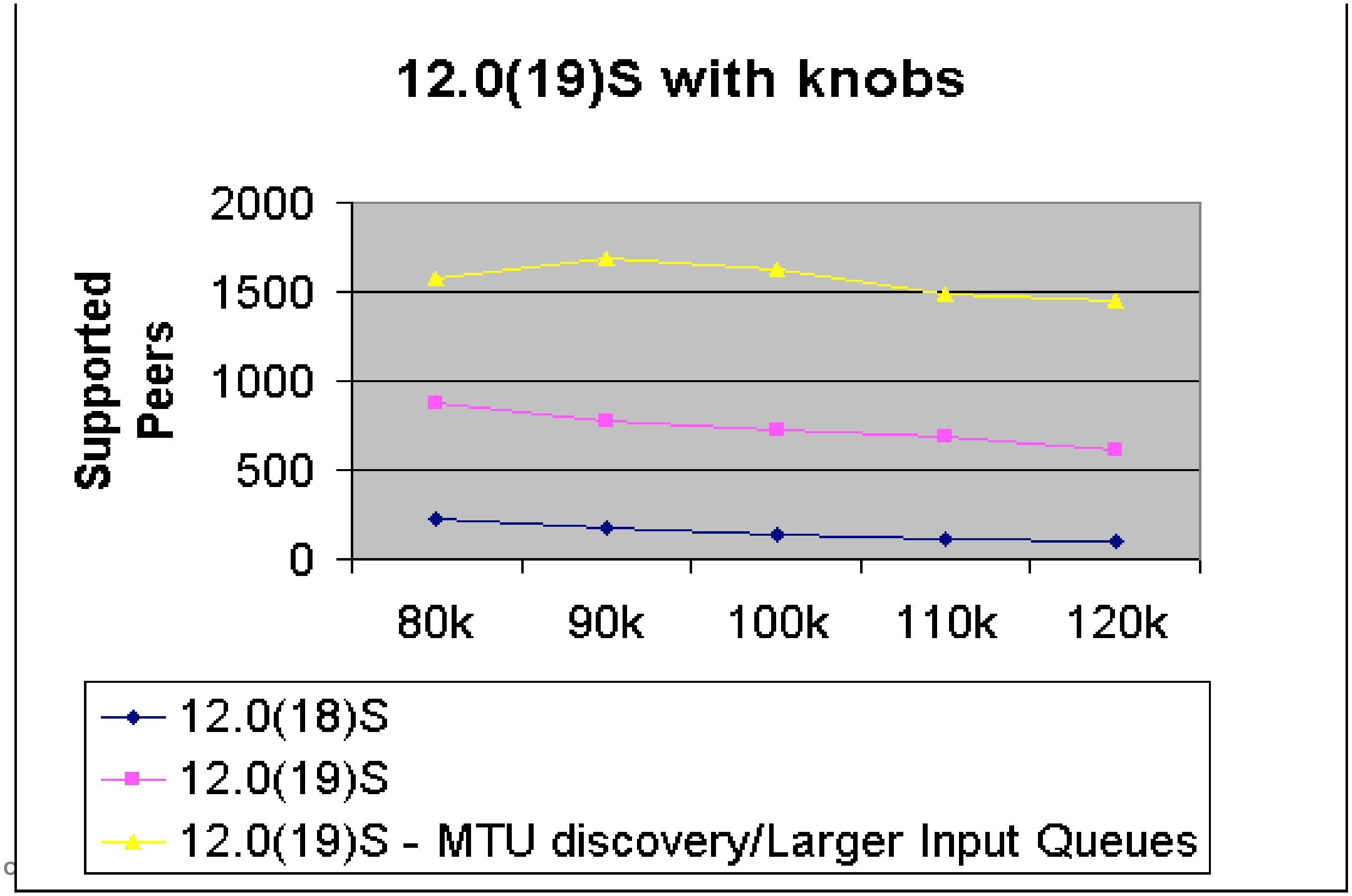
UPDATE Packing

- **4x – 6x improvement!!**



UPDATE Packing

- **12.0(19)S + MTU discovery + Larger Input Queues = 14x improvement**



Agenda

- Scalability
  - Protocol Issues
  - Initial Convergence
- Troubleshooting
  - Peer establishment
  - Missing Routes
  - Inconsistent Route Selection
  - Loops and Convergence Issues

Peer Establishment

- Routers establish a TCP session

  Port 179—permit in ACLs

  IP connectivity (route from IGP)

- OPEN messages are exchanged

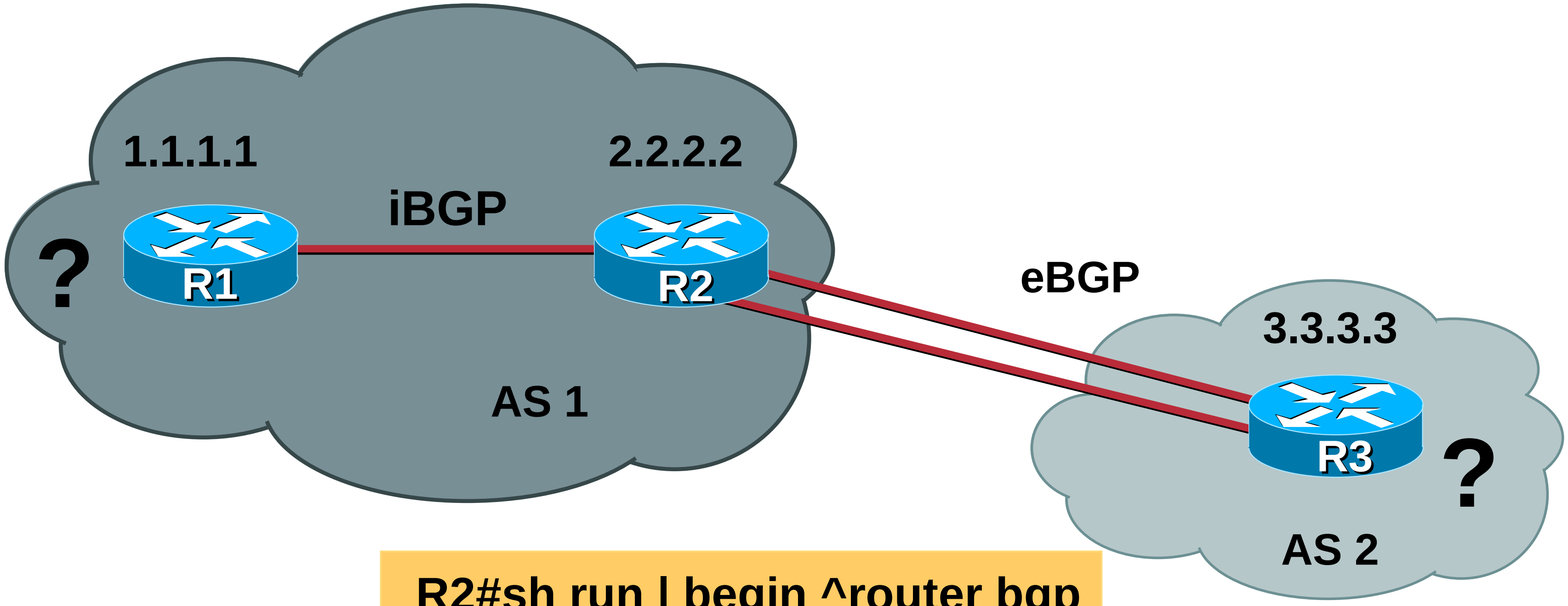
  Peering addresses must match the TCP session

  Local AS configuration parameters

Common Problems

- **Sessions are not established**
  - **No IP reachability**
  - **Incorrect configuration**
- **Peers are flapping**
  - **Layer 2 problems**

Peer Establishment - Diagram



```
R2#sh run | begin ^router bgp
router bgp 1
 bgp log-neighbor-changes
 neighbor 1.1.1.1 remote-as 1
 neighbor 3.3.3.3 remote-as 2
```

Peer Establishment - Symptoms

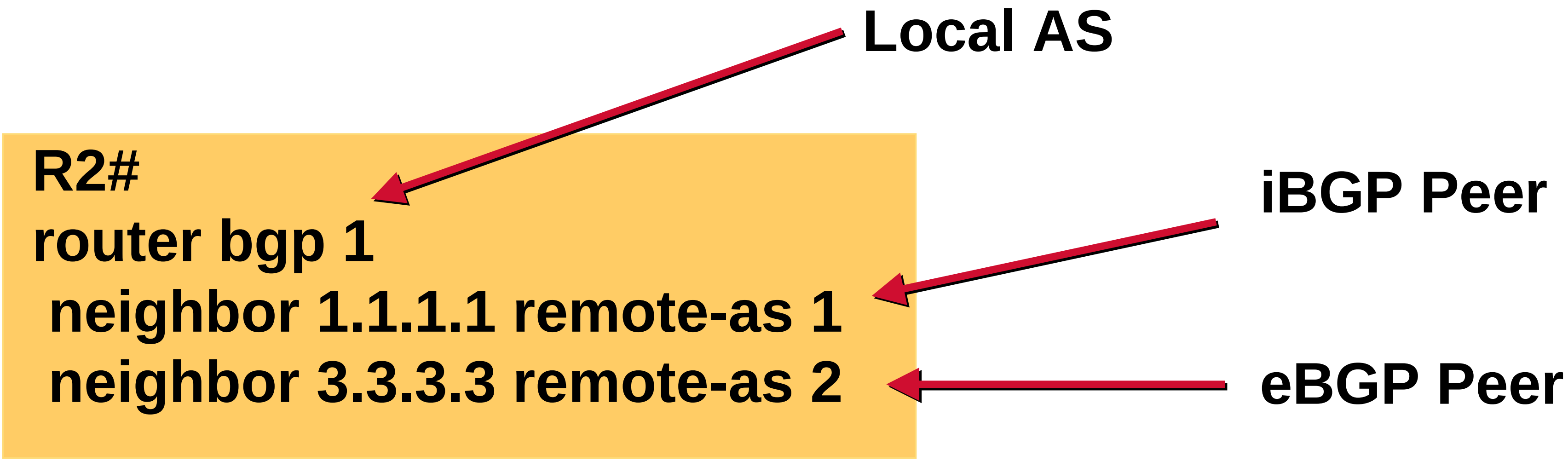
```
R2#show ip bgp summary
BGP router identifier 2.2.2.2, local AS number 1
BGP table version is 1, main routing table version 1
Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down State
1.1.1.1         4     1       0       0        0    0    0  never    Active
3.3.3.3         4     2       0       0        0    0    0  never    Idle
```

- **Both peers are having problems**

  State may change between Active, Idle and Connect

Peer Establishment

- Is the Local AS configured correctly?
- Is the remote-as assigned correctly?
- Verify with your diagram or other documentation!



```
R2#
router bgp 1
 neighbor 1.1.1.1 remote-as 1
 neighbor 3.3.3.3 remote-as 2
```

Peer Establishment - iBGP

- Assume that IP connectivity has been checked
- Check TCP to find out what connections we are accepting

```
R2#show tcp brief all
TCB          Local Address          Foreign Address       (state)
005F2934   *.179                    3.3.3.3.*              LISTEN
0063F3D4  *.179                     1.1.1.1.*              LISTEN
```

We are listening for TCP connections for port 179 for the configured peering addresses only!

```
R2#debug ip tcp transactions
TCP special event debugging is on
R2#
TCP: sending RST, seq 0, ack 2500483296
TCP: sent RST to 4.4.4.4:26385 from 2.2.2.2:179
```

Remote is trying to open the session from 4.4.4.4 address …

Peer Establishment - iBGP

What about us ?

```
R2#debug ip bgp
BGP debugging is on
R2#
BGP: 1.1.1.1 open active, local address 4.4.4.5
BGP: 1.1.1.1 open failed: Connection refused by remote host
```

We are trying to open the session from 4.4.4.5 address…

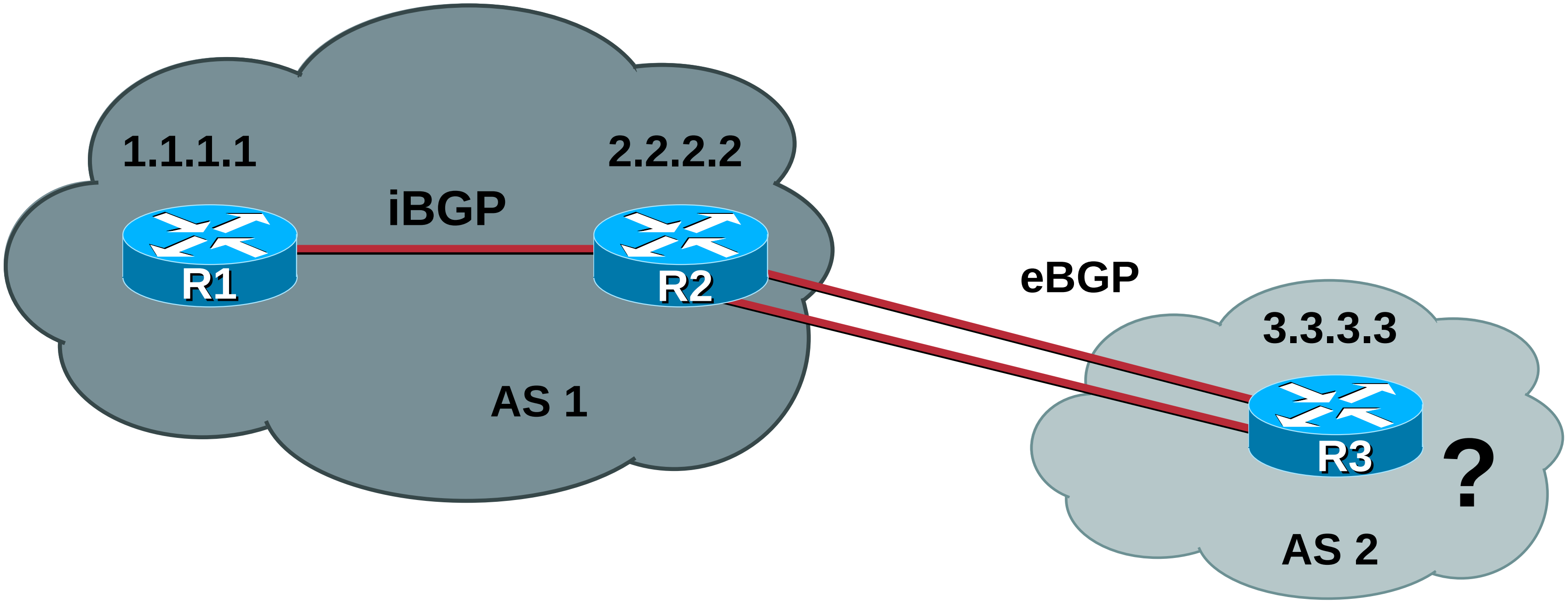
```
R2#sh ip route 1.1.1.1
Routing entry for 1.1.1.1/32
  Known via "static", distance 1, metric 0 (connected)
  * directly connected, via Serial1
      Route metric is 0, traffic share count is 1

R2#show ip interface brief | include Serial1
Serial1                 4.4.4.5       YES manual   up     up
```

Peer Establishment - iBGP

- Source address is the outgoing interface towards the destination but peering in this case is using loopback interfaces!
- Force both routers to source from the correct interface
- Use “update-source” to specify the loopback when loopback peering

```
R2#
router bgp 1
 neighbor 1.1.1.1 remote-as 1
 neighbor 1.1.1.1 update-source Loopback0
 neighbor 3.3.3.3 remote-as 2
 neighbor 3.3.3.3 update-source Loopback0
```

Peer Establishment - Diagram



- R1 is established now
- The eBGP session is still having trouble!

Peer Establishment - eBGP

- Trying to load-balance over multiple links to the eBGP peer
- Verify IP connectivity

Check the routing table

Use ping/trace to verify two way reachability

```
R2#ping 3.3.3.3
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 3.3.3.3, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/8 ms
```

- Routing towards destination correct, but…

Peer Establishment - eBGP

```
R2#ping ip
Target IP address: 3.3.3.3
Extended commands [n]: y
Source address or interface: 2.2.2.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 3.3.3.3, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

- **Use extended pings to test loopback to loopback connectivity**
- **R3 does not have a route to our loopback, 2.2.2.2**

Peer Establishment - eBGP

- **Assume R3 added a route to 2.2.2.2**
- **Still having problems…**

```
R2#sh ip bgp neigh 3.3.3.3
BGP neighbor is 3.3.3.3,  remote AS 2, external link
  BGP version 4, remote router ID 0.0.0.0
  BGP state = Idle
  Last read 00:00:04, hold time is 180, keepalive interval is 60 seconds
  Received 0 messages, 0 notifications, 0 in queue
  Sent 0 messages, 0 notifications, 0 in queue
  Route refresh request: received 0, sent 0
  Default minimum time between advertisement runs is 30 seconds
 For address family: IPv4 Unicast
  BGP table version 1, neighbor version 0
  Index 2, Offset 0, Mask 0x4
  0 accepted prefixes consume 0 bytes
  Prefix advertised 0, suppressed 0, withdrawn 0
  Connections established 0; dropped 0
  Last reset never
  External BGP neighbor not directly connected.
  No active TCP connection
```

Peer Establishment - eBGP

```
R2#
router bgp 1
 neighbor 3.3.3.3 remote-as 2
 neighbor 3.3.3.3 ebgp-multihop 255
 neighbor 3.3.3.3 update-source Loopback0
```

- eBGP peers are normally directly connected
  - By default, TTL is set to 1 for eBGP peers
  - If not directly connected, specify ebgp-multihop
- At this point, the session should come up

Peer Establishment - eBGP

```
R2#show ip bgp summary
BGP router identifier 2.2.2.2, local AS number 1

Neighbor    V   AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down   State/PfxRcd
3.3.3.3     4   2      10      26        0    0    0 never     Active
```

- **Still having trouble!**

  Connectivity issues have already been checked and corrected

Peer Establishment - eBGP

```
R2#debug ip bgp events
14:06:37: BGP: 3.3.3.3 open active, local address 2.2.2.2
14:06:37: BGP: 3.3.3.3 went from Active to OpenSent
14:06:37: BGP: 3.3.3.3 sending OPEN, version 4
14:06:37: BGP: 3.3.3.3 received NOTIFICATION 2/2
       (peer in wrong AS) 2 bytes 0001
14:06:37: BGP: 3.3.3.3 remote close, state CLOSEWAIT
14:06:37: BGP: service reset requests
14:06:37: BGP: 3.3.3.3 went from OpenSent to Idle
14:06:37: BGP: 3.3.3.3 closing
```

- **If an error is detected, a notification is sent and the session is closed**

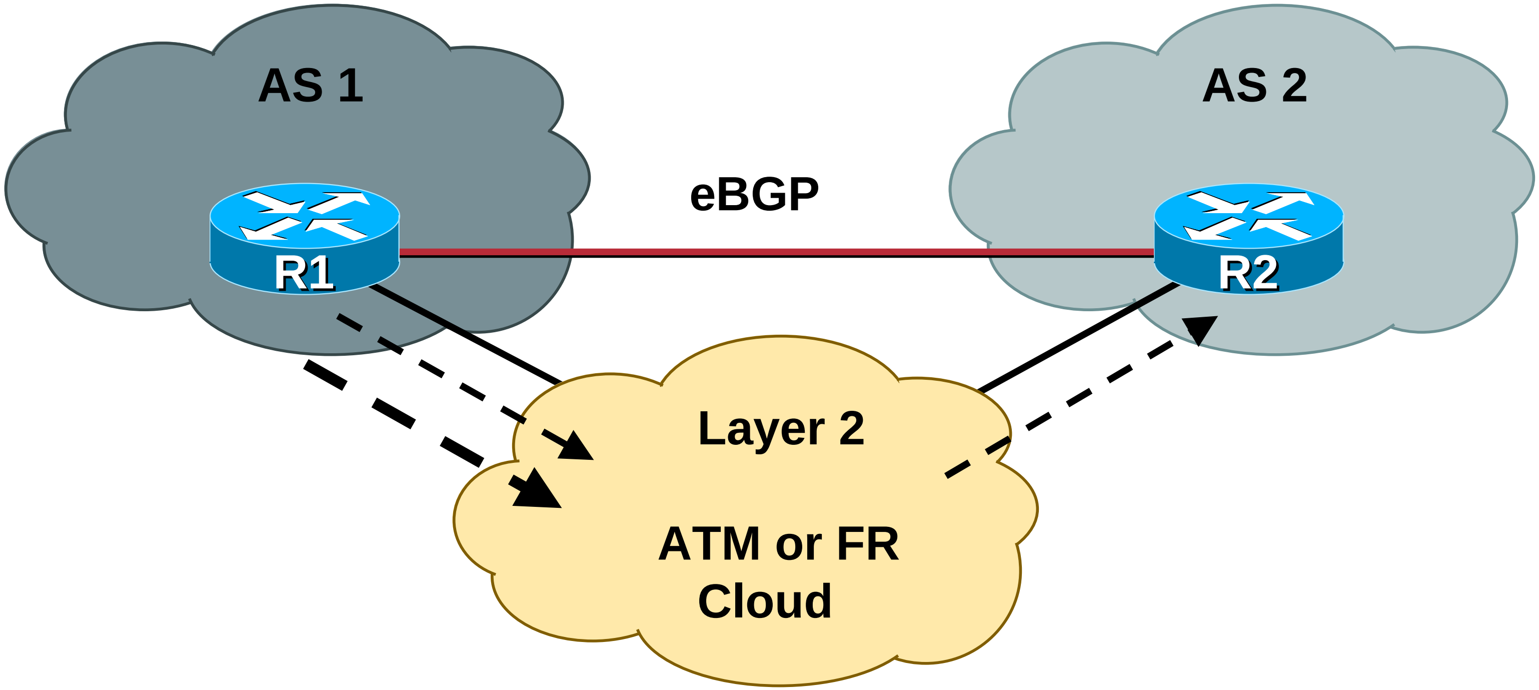
- **R3 is configured incorrectly**

  Has “neighbor 2.2.2.2 remote-as 10”

  Should have “neighbor 2.2.2.2 remote-as 1”

- **After R3 makes this correction the session comes up**

Flapping Peer - Diagram



- Small packets are ok
- Large packets are lost in the cloud
- BGP session flaps

Flapping Peer

- Enable “bgp log-neighbor-changes” so you get a log message when a peer flaps
- R1 and R2 are peering over ATM cloud

```
R2#

%BGP-5-ADJCHANGE: neighbor 1.1.1.1 Down BGP
  Notification sent

%BGP-3-NOTIFICATION: sent to neighbor 1.1.1.1 4/0
  (hold time expired) 0 bytes

R2#show ip bgp neighbor 1.1.1.1 | include Last reset

 Last reset 00:01:02, due to BGP Notification sent,
  hold time expired
```

- We are not receiving keepalives from the other side!

Flapping Peer

- Let’s take a look at our peer!

```
R1#show ip bgp sum
BGP router identifier 172.16.175.53, local AS number 1
BGP table version is 10167, main routing table version 10167
10166 network entries and 10166 paths using 1352078 bytes of memory
1 BGP path attribute entries using 60 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 activity 10166/300 prefixes, 10166/0 paths, scan interval 15 secs

Neighbor      V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2.2.2.2       4    2       53     284    10167    0   97 00:02:15        0
```

```
R1#show ip bgp summary | begin Neighbor
Neighbor      V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down  State/PfxRcd
2.2.2.2       4    2       53     284    10167    0   98 00:03:04        0
```

- Hellos are stuck in OutQ behind update packets!
- Notice that the MsgSent counter has not moved

Flapping Peer

```
R1#ping 2.2.2.2
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/21/24 m
```

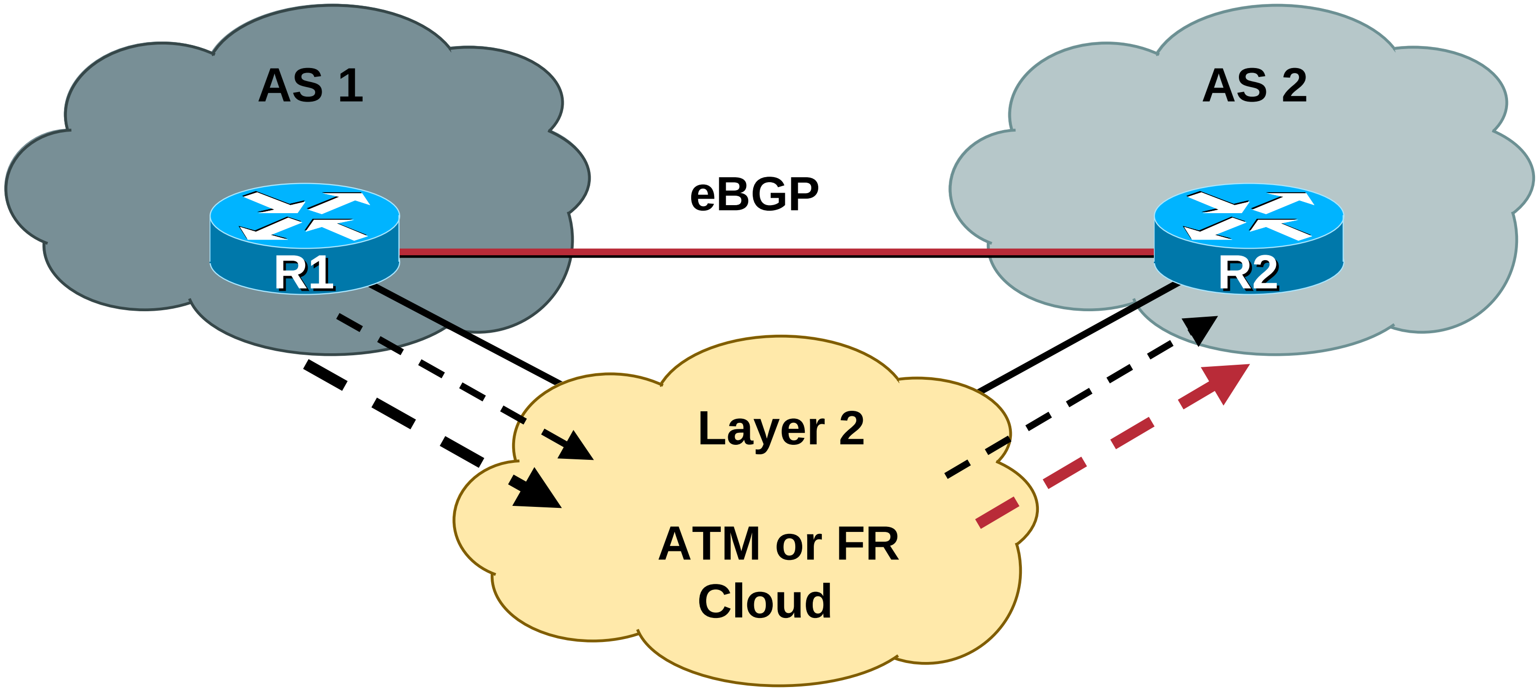
```
R1#ping ip
Target IP address: 2.2.2.2
Repeat count [5]:
Datagram size [100]: 1500
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 1500-byte ICMP Echos to 2.2.2.2, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

- **Normal pings work but a ping of 1500 fails?**

Flapping Peer

- Things to check
  - MTU values
  - Traffic shaping
  - Rate-limiting parameters
- Looks like a Layer 2 problem
- At this point we have verified that BGP is not at fault
- Next step is to troubleshoot layer 2…

Flapping Peer - Diagram



- - - ▶ Small Packets

▬ ▬ ▶ Large Packets

- Large packets are ok now
- BGP session is stable!

Agenda

- Scalability
  - Protocol Issues
  - Initial Convergence
- Troubleshooting
  - Peer establishment
  - Missing Routes
  - Inconsistent Route Selection
  - Loops and Convergence Issues

Quick Review

- Once sessions have been established
  - Bestpath algorithm selects best path for each prefix
  - Only the bestpath is advertised
  - UPDATE messages are used to exchange routes
- Incremental UPDATE messages are exchanged afterwards

Quick Review

- **Bestpath received from eBGP peer**

  Advertise to all peers

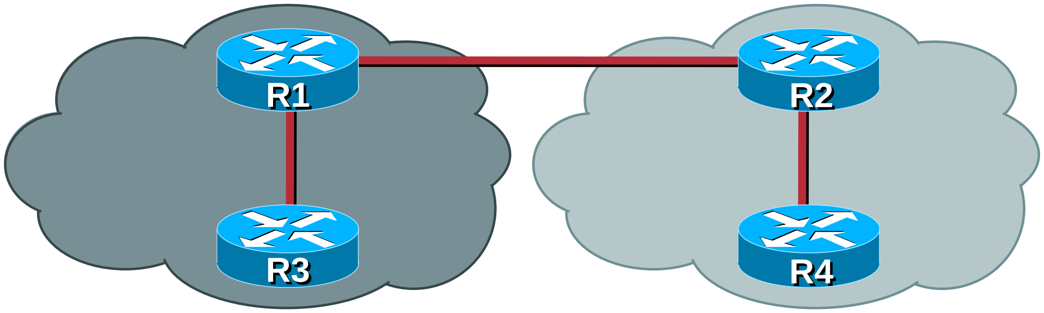
- **Bestpath received from iBGP peer**

  Advertise only to eBGP peers

  A full iBGP mesh must exist

Missing Routes - Example I

- Two RR clusters
- R1 is a RR for R3
- R2 is a RR for R4
- R4 is advertising 7.0.0.0/8
- R2 has the route but R1 and R3 do not?



Missing Routes - Example I

- First, did R2 advertise the route to R1?

```
R2# show ip bgp neighbors 1.1.1.1 advertised-routes
BGP table version is 2, local router ID is 2.2.2.2
   Network          Next Hop            Metric LocPrf Weight Path
*>i7.0.0.0        4.4.4.4              0    100      0 I
```

- Did R1 receive it?

```
R1# show ip bgp neighbors 2.2.2.2 routes
Total number of prefixes 0
```

Missing Routes - Example I

- Time to debug!!

```
access-list 100 permit ip host 7.0.0.0 host 255.0.0.0

R1# debug ip bgp update 100
```

- Tell R2 to resend his UPDATEs

```
R2# clear ip bgp 1.1.1.1 soft out
```

- R1 shows us something interesting

```
*Mar  1 21:50:12.410: BGP(0): 2.2.2.2 rcv UPDATE w/ attr:
nexthop 4.4.4.4, origin i, localpref 100, metric 0,
originator 100.1.1.1, clusterlist 2.2.2.2, path , community
, extended community

*Mar  1 21:50:12.410: BGP(0): 2.2.2.2 rcv UPDATE about
7.0.0.0/8 -- DENIED due to: ORIGINATOR is us;
```

- Cannot accept an update with our Router-ID as the ORIGINATOR_ID. Another means of loop detection in BGP

Missing Routes - Example I

- R1 and R4 have the same Router-ID

```
R1# show ip bgp summary | include identifier.
BGP router identifier 100.1.1.1, local AS number 100.
```

```
R4# show ip bgp summary | include identifier.
BGP router identifier 100.1.1.1, local AS number 100.
```

- Can be a problem in multicast networks; For RP (Rendezvous Point) purposes the same address may be assigned to multiple routers
- Specify a unique Router-ID

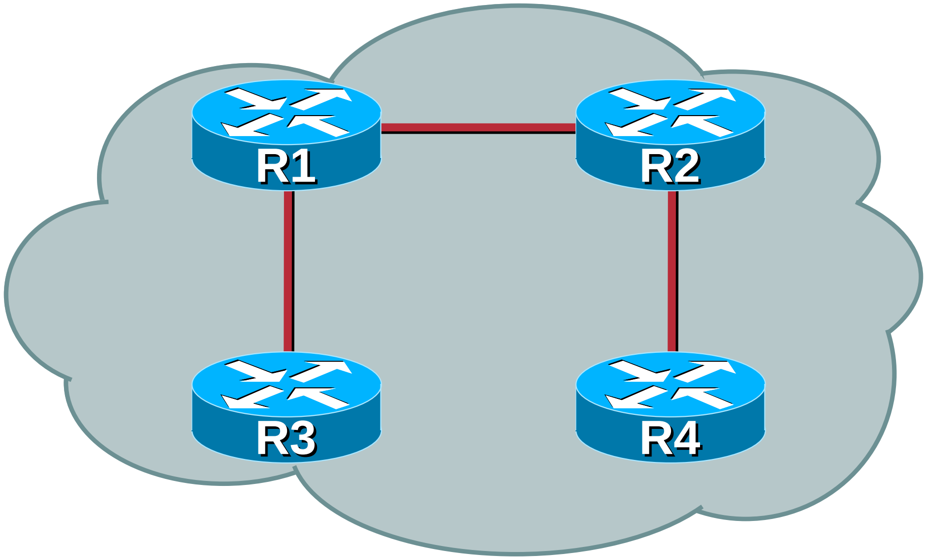
```
R1#show run | include router-id
 bgp router-id 1.1.1.1
R4# show run | include router-id
 bgp router-id 4.4.4.4
```

Missing Routes - Example II

- One RR cluster
- R1 and R2 are RRs
- R3 and R4 are RRCs
- R4 is advertising 7.0.0.0/8

  R2 has it

  R1 and R3 do not



```
R1#show run | include cluster
 bgp cluster-id 10
R2#show run | include cluster
 bgp cluster-id 10
```

Missing Routes - Example II

- Same steps as last time!
- Did R2 advertise it to R1?

```
R2# show ip bgp neighbors 1.1.1.1 advertised-routes
BGP table version is 2, local router ID is 2.2.2.2
Origin codes: i - IGP, e - EGP, ? - incomplete
   Network          Next Hop            Metric LocPrf Weight Path
*>i7.0.0.0        4.4.4.4                  0    100      0 i
```

- Did R1 receive it?

```
R1# show ip bgp neighbor 2.2.2.2 routes
Total number of prefixes 0
```

Missing Routes - Example II

- Time to debug!!

```
access-list 100 permit ip host 7.0.0.0 host 255.0.0.0

R1# debug ip bgp update 100
```

- Tell R2 to resend his UPDATEs

```
R2# clear ip bgp 1.1.1.1 soft out
```

- R1 shows us something interesting

```
*Mar  3 14:28:57.208: BGP(0): 2.2.2.2 rcv UPDATE w/ attr: nexthop
4.4.4.4, origin i, localpref 100, metric 0, originator 4.4.4.4,
clusterlist 0.0.0.10, path , community , extended community

*Mar  3 14:28:57.208: BGP(0): 2.2.2.2 rcv UPDATE about 7.0.0.0/8 --
DENIED due to: reflected from the same cluster;
```

- Remember, all RRCs must peer with all RRs in a cluster. Allows R4 to send the update directly to R1

Update Filtering

- Type of filters

  Prefix filters

  AS_PATH filters

  Community filters

  Route-maps

- Applied incoming and/or outgoing

Missing Routes - Update Filters

- Determine which filters are applied to the BGP session

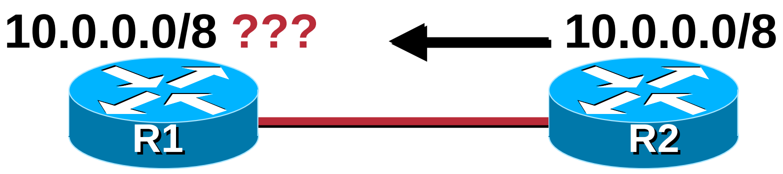
```
show ip bgp neighbors x.x.x.x
show run | include neighbor x.x.x.x
```

- Examine the route and pick out the relevant attributes

```
show ip bgp x.x.x.x
```

- Compare the attributes against the filters

Missing Routes - Update Filters



- Missing 10.0.0.0/8 in R1 (1.1.1.1)
- Not received from R2 (2.2.2.2)

```
R1#show ip bgp neigh 2.2.2.2 routes

Total number of prefixes 0
```

Missing Routes - Update Filters

- R2 originates the route
- Does not advertise it to R1

```
R2#show ip bgp neigh 1.1.1.1 advertised-routes
    Network          Next Hop            Metric LocPrf Weight Path
```

```
R2#show ip bgp 10.0.0.0
BGP routing table entry for 10.0.0.0/8, version 1660
Paths: (1 available, best #1)
  Not advertised to any peer
  Local
      0.0.0.0 from 0.0.0.0 (2.2.2.2)
      Origin IGP, metric 0, localpref 100, weight 32768, valid, sourced, local, best
```

Missing Routes - Update Filters

- Time to check filters!
- ^ matches the beginning of a line
- $ matches the end of a line
- ^$ means match any empty AS_PATH
- Filter “looks” correct

```
R2#show run | include neighbor 1.1.1.1
 neighbor 1.1.1.1 remote-as 3
 neighbor 1.1.1.1 filter-list 1 out

R2#sh ip as-path 1
  AS path access list 1
    permit ^$
```

Missing Routes - Update Filters

```
R2#show ip bgp filter-list 1

R2#show ip bgp regexp ^$
BGP table version is 1661, local router ID is 2.2.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i - internal
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.0.0.0         0.0.0.0              0          32768 i
```

- **Nothing matches the filter-list???**
- **Re-typing the regexp gives the expected output**

Missing Routes - Update Filters

- **Copy and paste** **the entire regexp line from the configuration**

```
R2#show ip bgp regexp  ^$
```

Nothing matches again! Let’s use the up arrow key to see where the cursor stops

```
R2#show ip bgp regexp  ^$  █
```

End of line is at the cursor

- **There is a trailing white space at the end**
- **It is considered part of the regular expression**

Missing Routes - Update Filters

- Force R2 to resend the update after the filter-list correction
- Then check R1 to see if he has the route

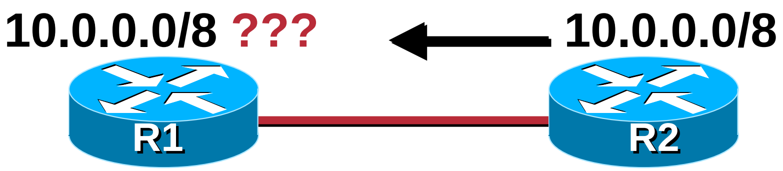
```
R2#clear ip bgp 1.1.1.1 soft out

R1#show ip bgp 10.0.0.0
% Network not in table
```

- R1 still does not have the route
- Time to check R1’s inbound policy for R2

Missing Routes - Update Filters

```
R1#show run | include neighbor 2.2.2.2
 neighbor 2.2.2.2 remote-as 12
 neighbor 2.2.2.2 route-map POLICY in
R1#show route-map POLICY
route-map POLICY, permit, sequence 10
  Match clauses:
    ip address (access-lists): 100 101
    as-path (as-path filter): 1
  Set clauses:
  Policy routing matches: 0 packets, 0 bytes
R1#show access-list 100
Extended IP access list 100
    permit ip host 10.0.0.0 host 255.255.0.0
R1#show access-list 101
Extended IP access list 101
    permit ip 200.1.0 0.0.0.255 host 255.255.255.0
R1#show ip as-path 1
AS path access list 1
    permit ^12$
```

Missing Routes - Update Filters



- **Confused? Let’s run some debugs**

```
R1#show access-list 99
Standard IP access list 99
    permit 10.0.0.0

R1#debug ip bgp 2.2.2.2 update 99
BGP updates debugging is on for access list 99 for neighbor 2.2.2.2

R1#
4d00h: BGP(0): 2.2.2.2 rcvd UPDATE w/ attr: nexthop 2.2.2.2, origin i,
 metric 0, path 12
4d00h: BGP(0): 2.2.2.2 rcvd 10.0.0.0/8 -- DENIED due to: route-map;
```

Missing Routes - Update Filters

```
R1#sh run | include neighbor 2.2.2.2
 neighbor 2.2.2.2 remote-as 12
 neighbor 2.2.2.2 route-map POLICY in
R1#sh route-map POLICY
route-map POLICY, permit, sequence 10
  Match clauses:
    ip address (access-lists): 100 101
    as-path (as-path filter): 1
  Set clauses:
  Policy routing matches: 0 packets, 0 bytes
R1#sh access-list 100
Extended IP access list 100
    permit ip host 10.0.0.0 host 255.255.0.0
R1#sh access-list 101
Extended IP access list 101
    permit ip 200.1.1.0 0.0.0.255 host 255.255.255.0
R1#sh ip as-path 1
AS path access list 1
    permit ^12$
```

Missing Routes - Update Filters

- Wrong mask! Needs to be /8 and the ACL allows a /16 only!

```
Extended IP access list 100
        permit ip host 10.0.0.0 host 255.255.0.0
```

- Should be

```
Extended IP access list 100
        permit ip host 10.0.0.0 host 255.0.0.0
```

- Use prefix-list instead, more difficult to make a mistake

```
ip prefix-list my_filter permit 10.0.0.0/8
```

- What about ACL 101?

  Multiple matches on the same line are ORed

  Multiple matches on different lines are ANDed

- ACL 101 does not matter because ACL 100 matches which satisfies the OR condition

Agenda

- **Scalability**
  - **Protocol Issues**
  - **Initial Convergence**
- **Troubleshooting**
  - **Peer establishment**
  - **Missing Routes**
  - **Inconsistent Route Selection**
  - **Loops and Convergence Issues**

Inconsistent Route Selection

- Two common problems with route selection
  - Inconsistency
  - Appearance of an incorrect decision
- RFC 1771 defines the decision algorithm
- Every vendor has tweaked the algorithm
  - http://www.cisco.com/warp/public/459/25.shtml
- Route selection problems can result from oversights by RFC 1771

Inconsistent - Example I

- RFC says that MED is not always compared
- As a result, the ordering of the paths can effect the decision process
- By default, the prefixes are compared in order of arrival (most recent to oldest)
    - Use bgp deterministic-med to order paths consistently
    - The bestpath is recalculated as soon as the command is entered
    - Enable in all the routers in the AS

Inconsistent - Example I

- Inconsistent route selection may cause problems

  Routing loops

  Convergence loops – i.e. the protocol continuously sends updates in an attempt to converge

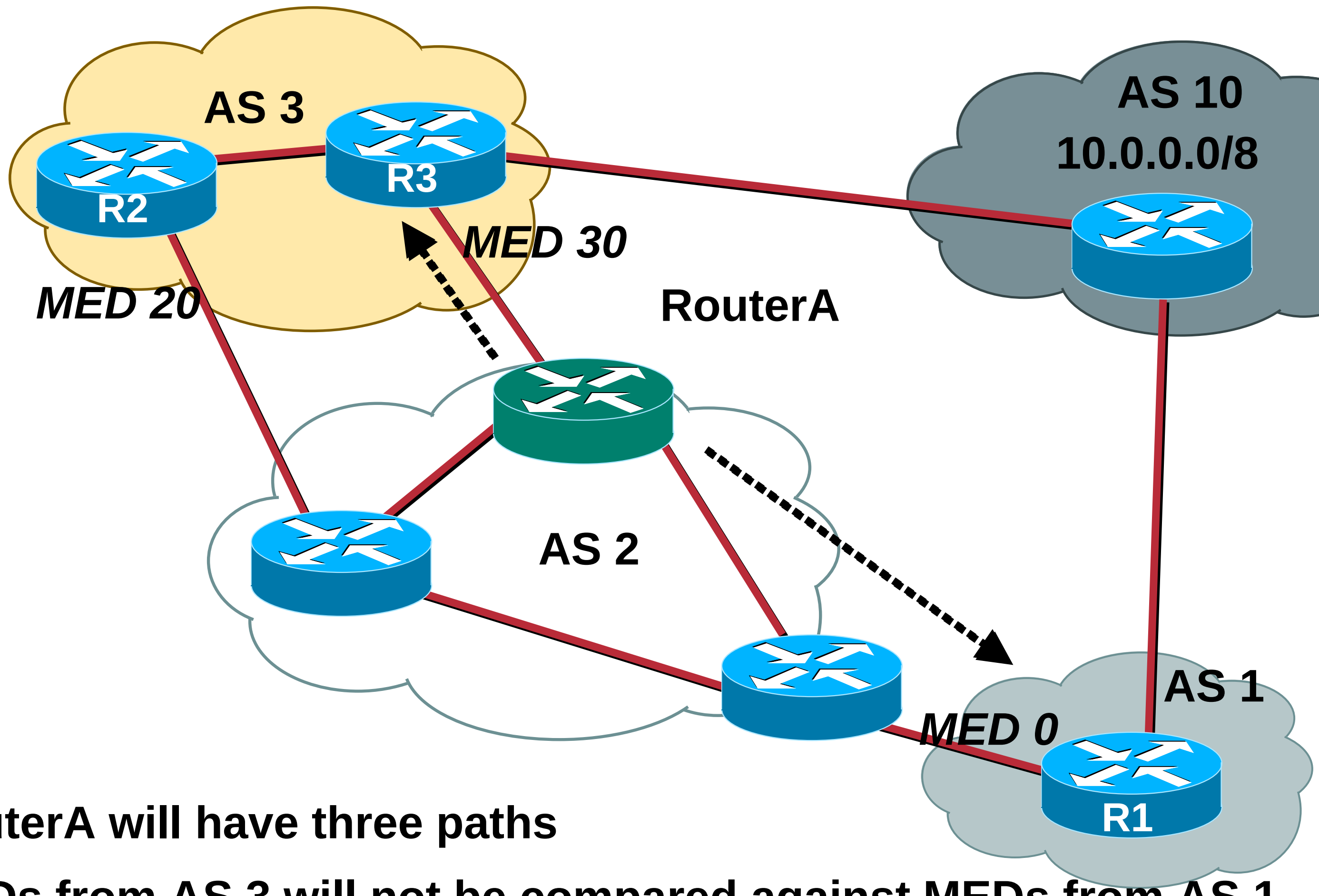
  Changes in traffic patterns

- Difficult to catch and troubleshoot

- It is best to avoid the problem in the first place

  bgp deterministic-med

Symptom I - Diagram



- RouterA will have three paths
- MEDs from AS 3 will not be compared against MEDs from AS 1
- RouterA will sometimes select the path from R1 as best and but may also select the path from R3 as best

Inconsistent - Example I

```
RouterA#sh ip bgp 10.0.0.0
BGP routing table entry for 10.0.0.0/8, version 40
Paths: (3 available, best #3, advertised over iBGP, eBGP)
  3 10
    2.2.2.2 from 2.2.2.2
      Origin IGP, metric 20, localpref 100, valid, internal
  3 10
    3.3.3.3 from 3.3.3.3
      Origin IGP, metric 30, valid, external
  1 10
    1.1.1.1 from 1.1.1.1
      Origin IGP, metric 0, localpref 100, valid, internal, best
```

- **Initial State**

  Path 1 beats Path 2—Lower MED

  Path 3 beats Path 1—Lower Router-ID

Inconsistent - Example I

```
RouterA#sh ip bgp 10.0.0.0
BGP routing table entry for 10.0.0.0/8, version 40
Paths: (3 available, best #3, advertised over iBGP, eBGP)
  1 10
    1.1.1.1 from 1.1.1.1
      Origin IGP, metric 0, localpref 100, valid, internal
  3 10
    2.2.2.2 from 2.2.2.2
      Origin IGP, metric 20, localpref 100, valid, internal
  3 10
    3.3.3.3 from 3.3.3.3
      Origin IGP, metric 30, valid, external, best
```

- **1.1.1.1 bounced so the paths are re-ordered**

  Path 1 beats Path 2—Lower Router-ID

  Path 3 beats Path 1—External vs Internal

Deterministic MED - Operation

- The paths are ordered by Neighbor AS
- The bestpath for each Neighbor AS group is selected
- The overall bestpath results from comparing the winners from each group
- The bestpath will be consistent because paths will be placed in a deterministic order

Deterministic MED - Result

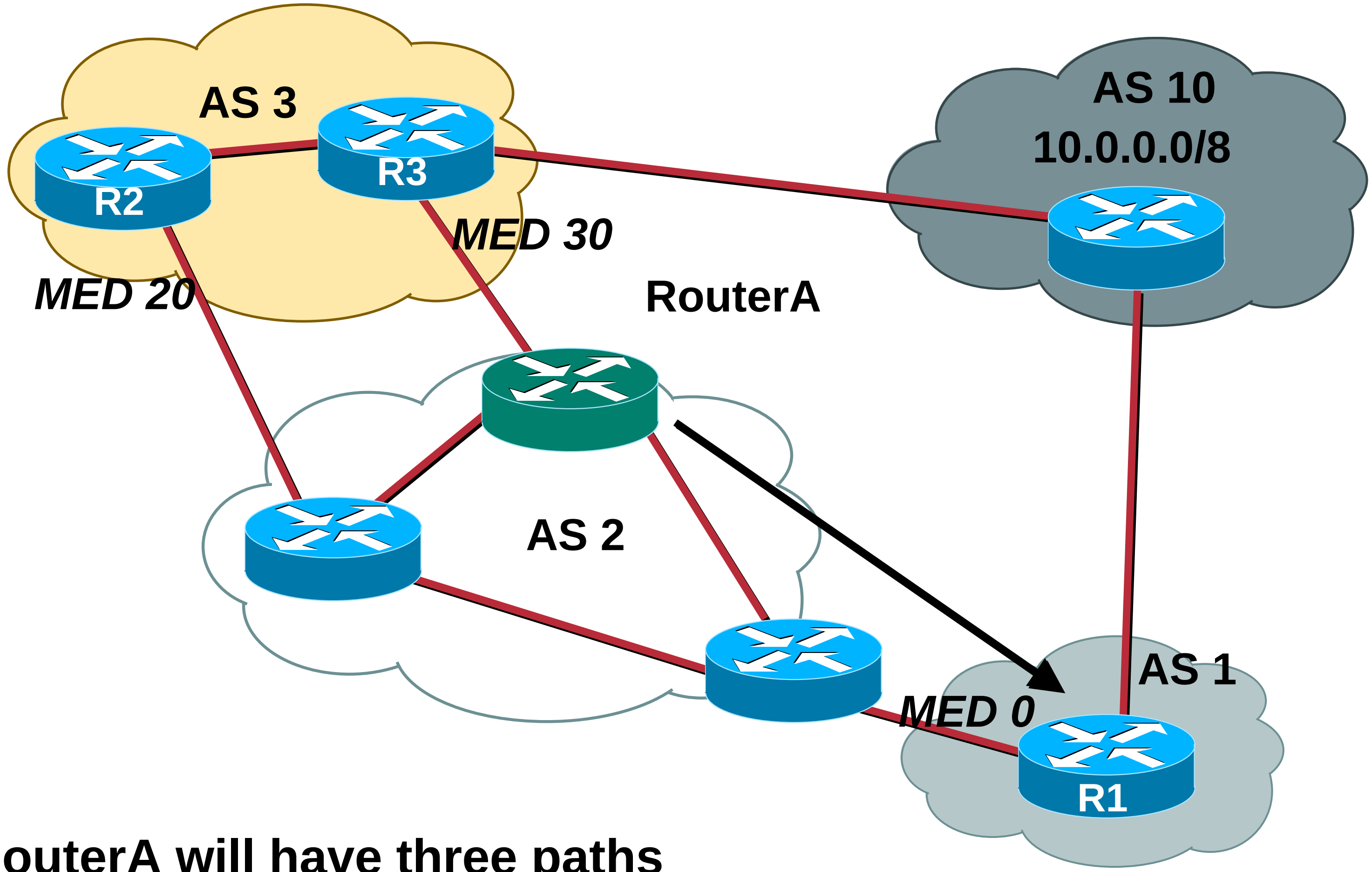
```
RouterA#sh ip bgp 10.0.0.0
BGP routing table entry for 10.0.0.0/8, version 40
Paths: (3 available, best #1, advertised over iBGP, eBGP)
  1 10
    1.1.1.1 from 1.1.1.1
      Origin IGP, metric 0, localpref 100, valid, internal, best
  3 10
    2.2.2.2 from 2.2.2.2
      Origin IGP, metric 20, localpref 100, valid, internal
  3 10
    3.3.3.3 from 3.3.3.3
      Origin IGP, metric 30, valid, external
```

Path 1 is best for AS 1

Path 2 beats Path 3 for AS 3—Lower MED

Path 1 beats Path 2—Lower Router-ID

Solution – Diagram



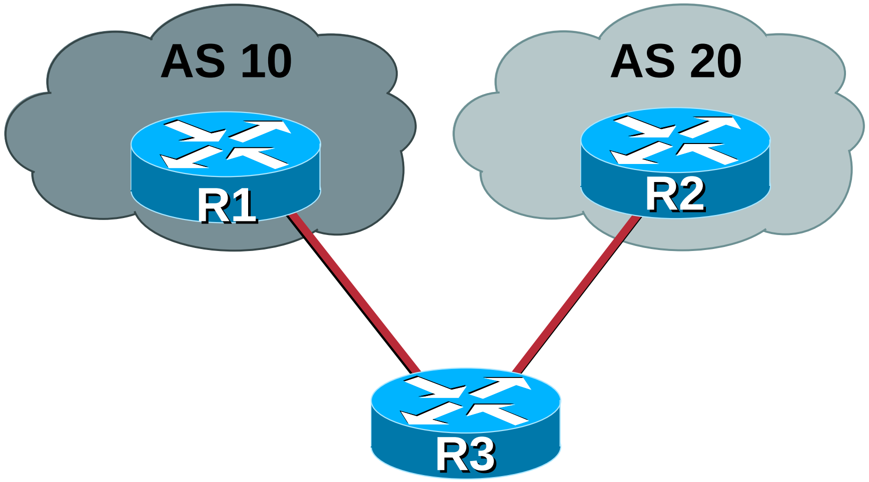
- RouterA will have three paths
- RouterA will consistently select the path from R1 as best!

Deterministic MED - Summary

- Always use "bgp deterministic-med"
- Need to enable throughout entire network at roughly the same time
- If only enabled on a portion of the network routing loops and/or convergence problems may become more severe
- As a result, default behavior cannot be changed so the knob must be configured by the user

Inconsistent - Example II

- **The bestpath changes every time the peering is reset**



```
R3#show ip bgp 7.0.0.0
BGP routing table entry for 7.0.0.0/8, version 15
  10 100
    1.1.1.1 from 1.1.1.1
      Origin IGP, metric 0, localpref 100, valid, external
  20 100
    2.2.2.2 from 2.2.2.2
      Origin IGP, metric 0, localpref 100, valid, external, best
```

Inconsistent - Example II

```
R3#show ip bgp 7.0.0.0
BGP routing table entry for 7.0.0.0/8, version 17
Paths: (2 available, best #2)
  Not advertised to any peer
  20 100
    2.2.2.2 from 2.2.2.2
      Origin IGP, metric 0, localpref 100, valid, external
  10 100
    1.1.1.1 from 1.1.1.1
      Origin IGP, metric 0, localpref 100, valid, external, best
```

- **The “oldest” external is the bestpath**
  - **All other attributes are the same**
  - **Stability enhancement!!—CSCdk12061—Integrated in 12.0(1)**
- **“bgp bestpath compare-router-id” will disable this enhancement—CSCdr47086—Integrated in 12.0(11)S and 12.1(3)**

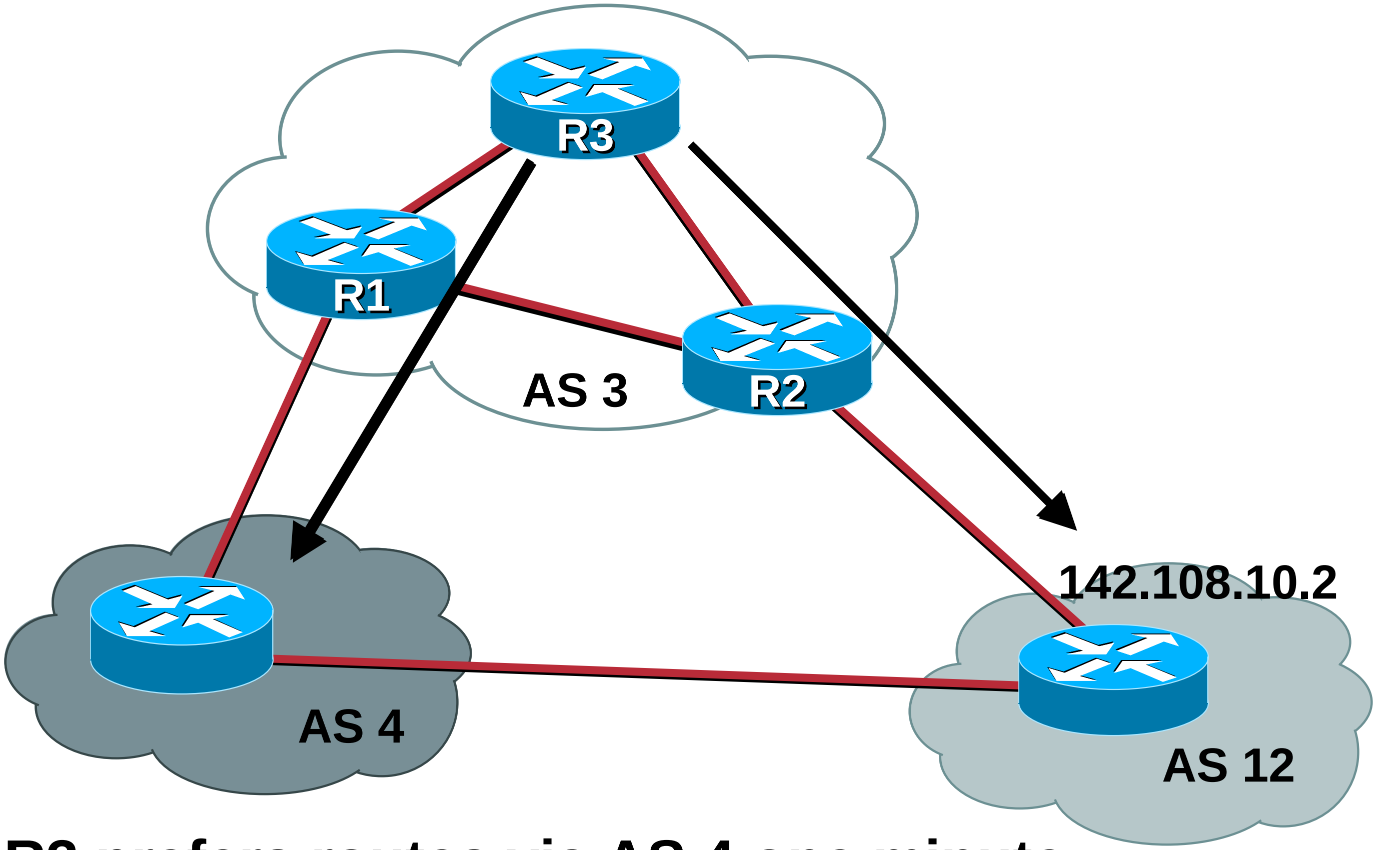
Agenda

- Scalability
    - Protocol Issues
    - Initial Convergence
- Troubleshooting
    - Peer establishment
    - Missing Routes
    - Inconsistent Route Selection
    - Loops and Convergence Issues

Route Oscillation

- One of the most common problems!
- Every minute routes flap in the routing table from one nexthop to another
- With full routes the most obvious symptom is high CPU in “BGP Router” process

Route Oscillation - Diagram



- R3 prefers routes via AS 4 one minute
- BGP scanner runs then R3 prefers routes via AS 12
- The entire table oscillates every 60 seconds

Route Oscillation - Symptom

```
R3#show ip bgp summary
BGP router identifier 3.3.3.3, local AS number 3
BGP table version is 502, main routing table version 502
267 network entries and 272 paths using 34623 bytes of memory

R3#sh ip route summary | begin bgp
bgp 3          4          6              520        1400
  External: 0 Internal: 10 Local: 0
internal       5                                    5800
Total          10         263         13936        43320
```

- **Watch for:**

  Table version number incrementing rapidly

  Number of networks/paths or external/internal routes changing

Route Oscillation - Troubleshooting

- Pick a route from the RIB that has changed within the last minute
- Monitor that route to see if it changes every minute

```
R3#show ip route 156.1.0.0
Routing entry for 156.1.0.0/16
  Known via "bgp 3", distance 200, metric 0
 Routing Descriptor Blocks:
  * 1.1.1.1, from 1.1.1.1, 00:00:53 ago
      Route metric is 0, traffic share count is 1
      AS Hops 2, BGP network version 474
```

```
R3#show ip bgp 156.1.0.0
BGP routing table entry for 156.1.0.0/16, version 474
Paths: (2 available, best #1)
  Advertised to non peer-group peers:
    2.2.2.2
  4 12
    1.1.1.1 from 1.1.1.1 (1.1.1.1)
      Origin IGP, localpref 100, valid, internal, best
  12
    142.108.10.2 (inaccessible) from 2.2.2.2 (2.2.2.2)
      Origin IGP, metric 0, localpref 100, valid, internal
```

Route Oscillation - Troubleshooting

- Check again after bgp_scanner runs
- bgp_scanner runs every 60 seconds and validates reachability to all nexthops

```
R3#sh ip route 156.1.0.0
Routing entry for 156.1.0.0/16
  Known via "bgp 3", distance 200, metric 0
    Routing Descriptor Blocks:
  * 142.108.10.2, from 2.2.2.2, 00:00:27 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1, BGP network version 478

R3#sh ip bgp 156.1.0.0
BGP routing table entry for 156.1.0.0/16, version 478
Paths: (2 available, best #2)
  Advertised to non peer-group peers:
    1.1.1.1
  4 12
    1.1.1.1 from 1.1.1.1 (1.1.1.1)
      Origin IGP, localpref 100, valid, internal
  12
    142.108.10.2 from 2.2.2.2 (2.2.2.2)
      Origin IGP, metric 0, localpref 100, valid, internal, best
```

Route Oscillation - Troubleshooting

- Lets take a closer look at the nexthop

```
R3#show ip route 142.108.10.2
Routing entry for 142.108.0.0/16
  Known via "bgp 3", distance 200, metric 0
 Routing Descriptor Blocks:
  * 142.108.10.2, from 2.2.2.2, 00:00:50 ago
      Route metric is 0, traffic share count is 1
      AS Hops 1, BGP network version 476

R3#show ip bgp 142.108.10.2
BGP routing table entry for 142.108.0.0/16, version 476
Paths: (2 available, best #2)
  Advertised to non peer-group peers:
    1.1.1.1
  4 12
    1.1.1.1 from 1.1.1.1 (1.1.1.1)
      Origin IGP, localpref 100, valid, internal
  12
    142.108.10.2 from 2.2.2.2 (2.2.2.2)
      Origin IGP, metric 0, localpref 100, valid, internal, best
```

Route Oscillation - Troubleshooting

- BGP nexthop is known via BGP
- Illegal recursive lookup
- Scanner will notice and install the other path in the RIB

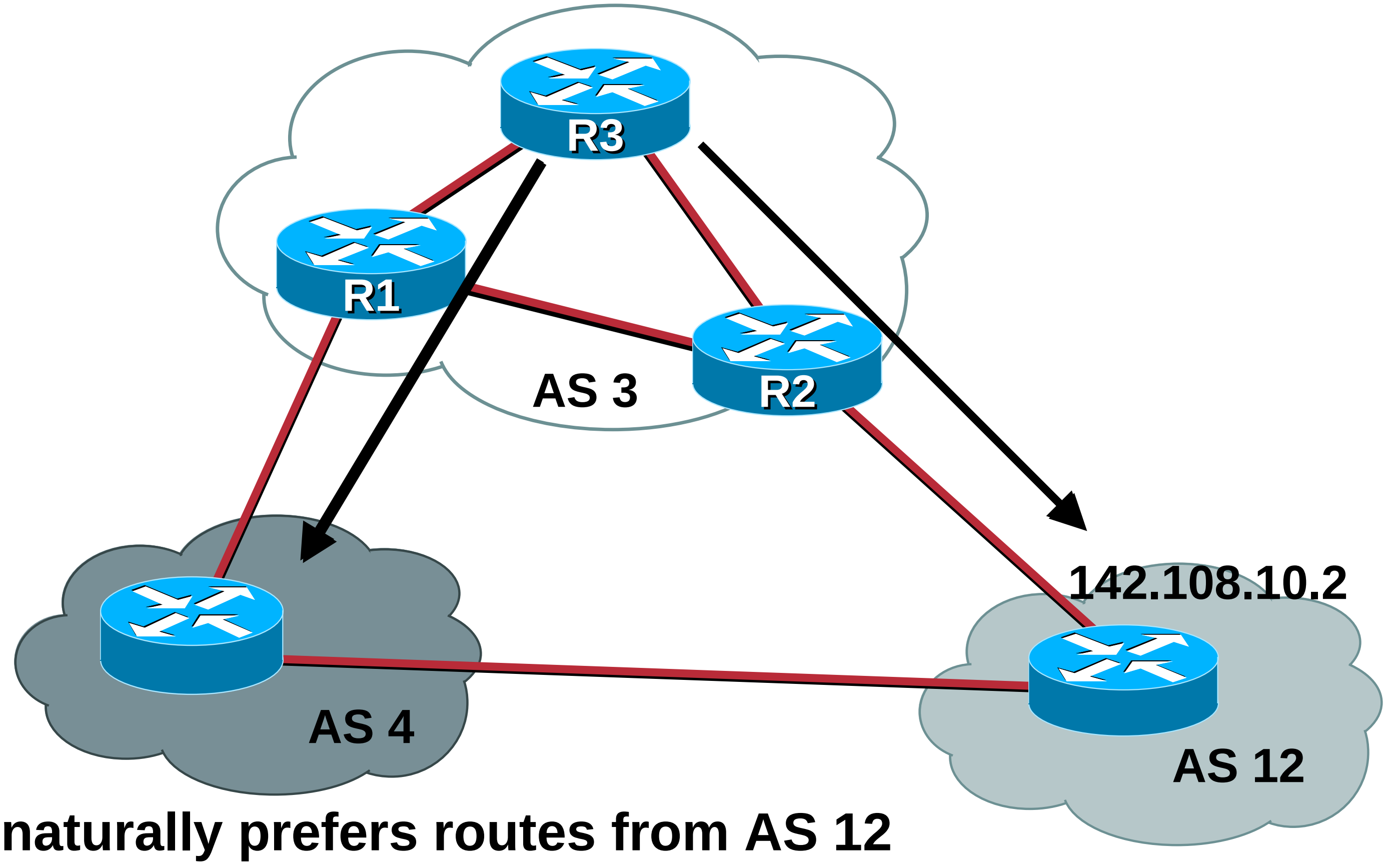
```
R3#sh debug
  BGP events debugging is on
  BGP updates debugging is on
  IP routing debugging is on
R3#
BGP: scanning routing tables
BGP: nettable_walker 142.108.0.0/16 calling revise_route
RT: del 142.108.0.0 via 142.108.10.2, bgp metric [200/0]
BGP: revise route installing 142.108.0.0/16 -> 1.1.1.1
RT: add 142.108.0.0/16 via 1.1.1.1, bgp metric [200/0]
RT: del 156.1.0.0 via 142.108.10.2, bgp metric [200/0]
BGP: revise route installing 156.1.0.0/16 -> 1.1.1.1
RT: add 156.1.0.0/16 via 1.1.1.1, bgp metric [200/0]
```

Route Oscillation - Troubleshooting

- Route to the nexthop is now valid
- Scanner will detect this and re-install the other path
- Routes will oscillate forever

```
R3#
BGP: scanning routing tables
BGP:  ip nettable_walker 142.108.0.0/16 calling revise_route
RT: del 142.108.0.0 via 1.1.1.1, bgp metric [200/0]
BGP: revise route installing 142.108.0.0/16 -> 142.108.10.2
RT: add 142.108.0.0/16 via 142.108.10.2, bgp metric [200/0]
BGP: nettable_walker 156.1.0.0/16 calling revise_route
RT: del 156.1.0.0 via 1.1.1.1, bgp metric [200/0]
BGP: revise route installing 156.1.0.0/16 -> 142.108.10.2
RT: add 156.1.0.0/16 via 142.108.10.2, bgp metric [200/0]
```

Route Oscillation – Step by step



- R3 naturally prefers routes from AS 12
- R3 does not have an IGP route to 142.108.10.2 which is the next-hop for routes learned via AS 12
- R3 learns 142.108.0.0/16 via AS 4 so 142.108.10.2 becomes reachable

Route Oscillation – Step by step

- R3 then prefers the AS 12 route for 142.108.0.0/16 whose next-hop is 142.108.10.2
- This is an illegal recursive lookup
- BGP detects the problem when scanner runs and flags 142.108.10.2 as inaccessible
- Routes through AS 4 are now preferred
- The cycle continues forever…

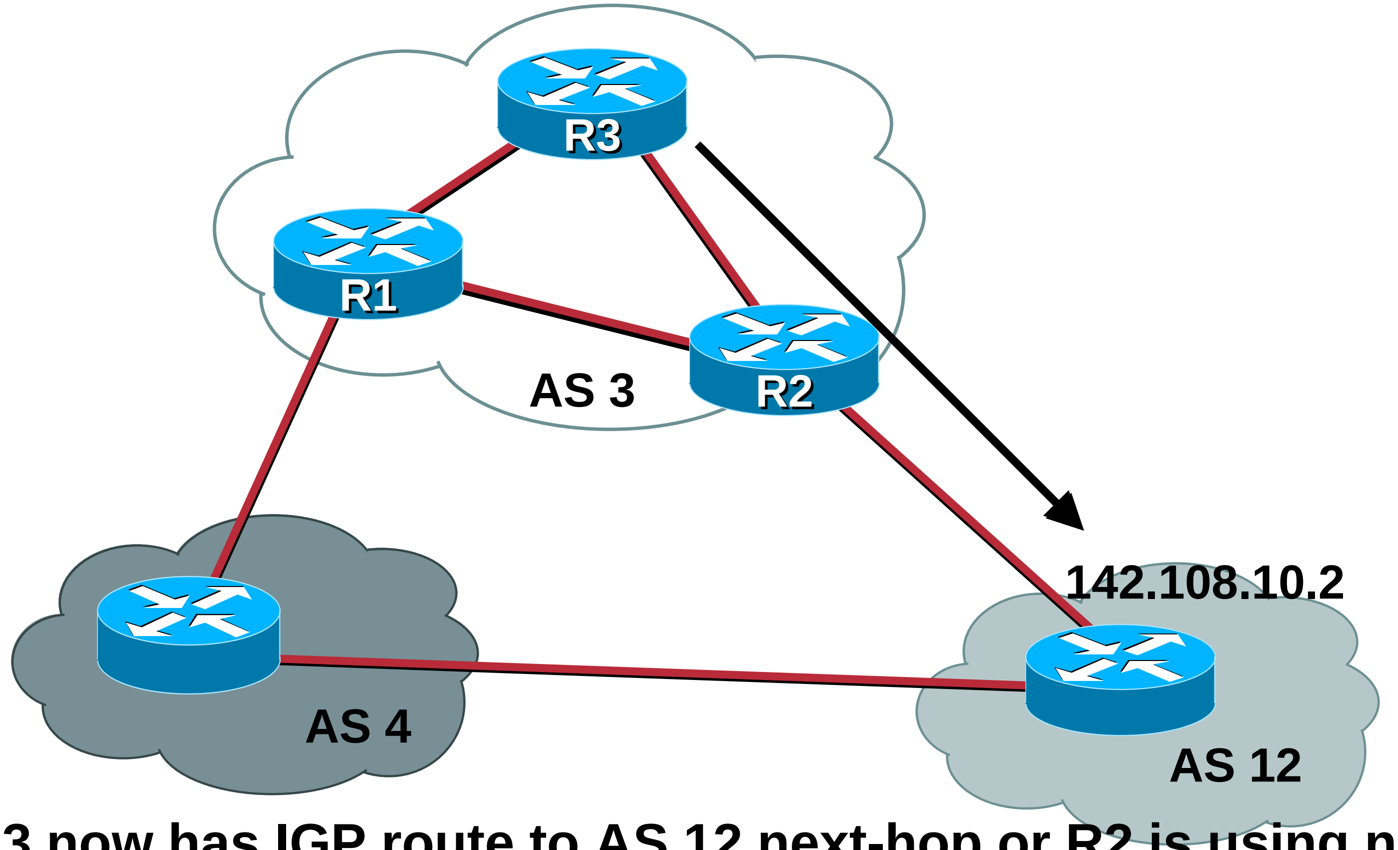
Route Oscillation – Solution

- iBGP preserves the next-hop information from eBGP
- To avoid problems

  Use “next-hop-self” for iBGP peering

  Make sure you advertise the next-hop prefix via the IGP

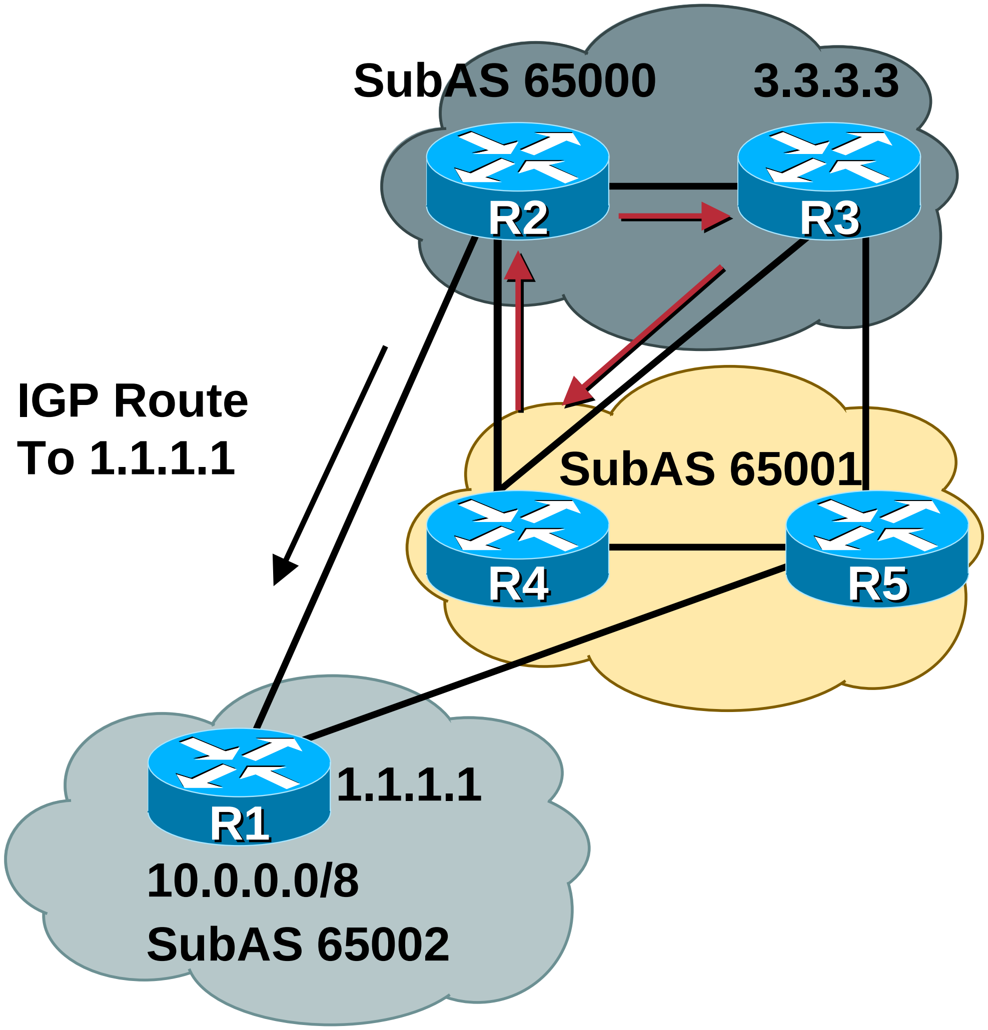
Route Oscillation - Solution



- R3 now has IGP route to AS 12 next-hop or R2 is using next-hop-self
- R3 now prefers routes via AS 12 all the time
- No more oscillation!!

Routing Loop

```
R5# traceroute 10.1.1.1

  1 30.100.1.1
  2 20.20.20.4  - R3
  3 30.1.1.26    - R4
  4 30.1.1.17    - R2
  5 20.20.20.4  - R3
  6 30.1.1.26    - R4
  7 30.1.1.17    - R2
  8 20.20.20.4
  9 30.1.1.26
 10 30.1.1.17
```



- **Traffic loops between R3, R4, and R2**

Routing Loop

- **First capture a “show ip route” from the three problem routers**
- **R3 is forwarding traffic to 1.1.1.1 (R1)**

```
R3# show ip route 10.1.1.1
Routing entry for 10.0.0.0/8
  Known via "bgp 65000", distance 200, metric 0
  Routing Descriptor Blocks:
    1.1.1.1, from 5.5.5.5, 01:46:43 ago
      Route metric is 0, traffic share count is 1
      AS Hops 0, BGP network version 0
  * 1.1.1.1, from 4.4.4.4, 01:46:43 ago
      Route metric is 0, traffic share count is 1
      AS Hops 0, BGP network version 0
```

Routing Loop

- **R4 is also forwarding to 1.1.1.1 (R1)**

```
R4# show ip route 10.1.1.1
Routing entry for 10.0.0.0/8
  Known via "bgp 65001", distance 200, metric 0
  Routing Descriptor Blocks:
  * 1.1.1.1, from 5.5.5.5, 01:47:02 ago
      Route metric is 0, traffic share count is 1
      AS Hops 0
```

Routing Loop

- R2 is forwarding to 3.3.3.3? (R3)

```
R2# show ip route 10.1.1.1
Routing entry for 10.0.0.0/8
  Known via "bgp 65000", distance 200, metric 0
Routing Descriptor Blocks:
  * 3.3.3.3, from 3.3.3.3, 01:47:00 ago
      Route metric is 0, traffic share count is 1
      AS Hops 0, BGP network version 3
```

- Very odd that the NEXT_HOP is in the middle of the network

Routing Loop

- **Verify BGP paths on R2**

```
R2#show ip bgp 10.0.0.0
BGP routing table entry for 10.0.0.0/8, version 3
Paths: (4 available, best #1)
  Advertised to non peer-group peers:
    1.1.1.1 5.5.5.5 4.4.4.4
  (65001 65002)
    3.3.3.3 (metric 11) from 3.3.3.3 (3.3.3.3)
      Origin IGP, metric 0, localpref 100, valid, confed-internal,
best
 (65002)
    1.1.1.1 (metric 50) from 1.1.1.1 (1.1.1.1)
      Origin IGP, metric 0, localpref 100, valid, confed-external
```

- **R3 path is better than R1 path because of IGP cost to the NEXT_HOP**
- **R3 is advertising the path to us with a NEXT_HOP of 3.3.3.3 ???**

Routing Loop

- **What is R3 advertising?**

```
R3# show ip bgp 10.0.0.0
BGP routing table entry for 10.0.0.0/8, version 3
Paths: (2 available, best #1, table Default-IP-Routing-Table)
  Advertised to non peer-group peers:
  5.5.5.5 2.2.2.2
  (65001 65002)
    1.1.1.1 (metric 5031) from 4.4.4.4 (4.4.4.4)
      Origin IGP, metric 0, localpref 100, valid, confed-external, best, multipath
  (65001 65002)
    1.1.1.1 (metric 5031) from 5.5.5.5 (5.5.5.5)
      Origin IGP, metric 0, localpref 100, valid, confed-external, multipath
```

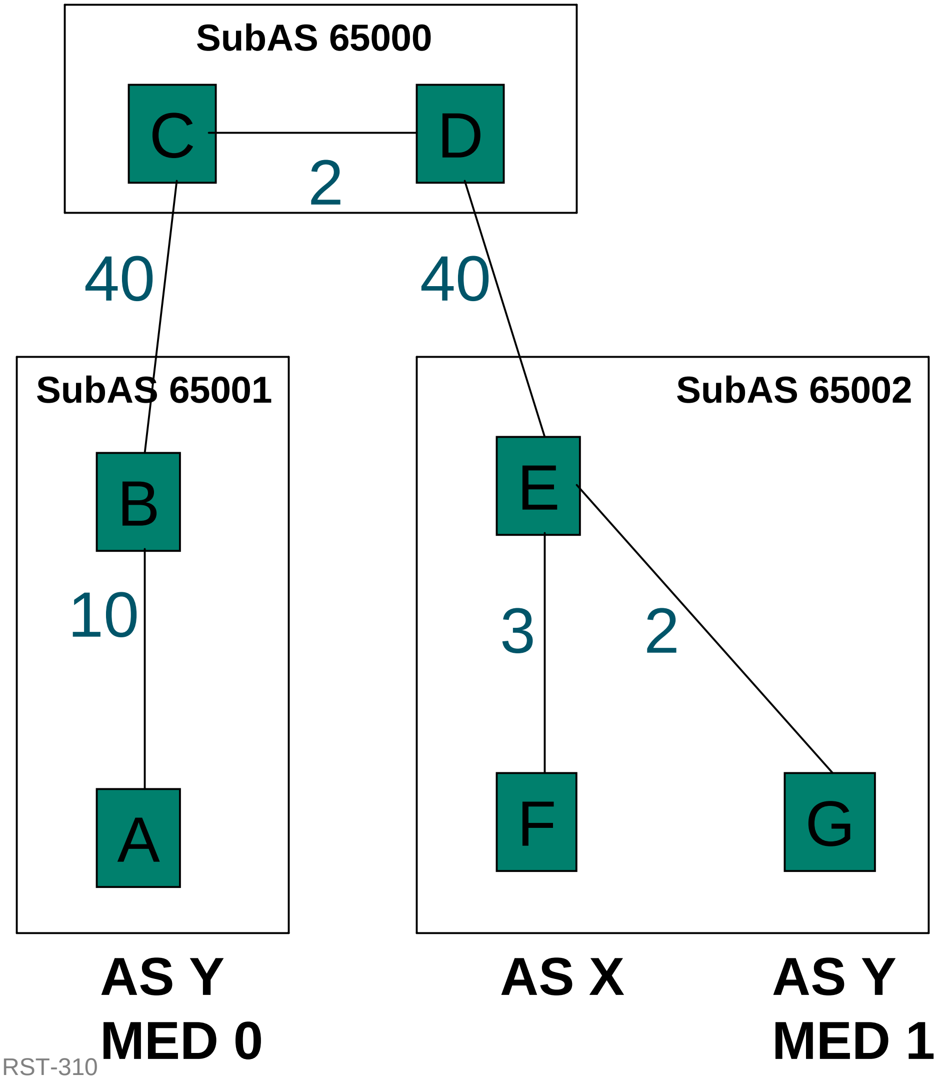
- **Hmmm, R3 is using multipath to load-balance**

```
R3#show run | i maximum
 maximum-paths 6
```

Routing Loop

- “maximum-paths” tells the router to reset the NEXT_HOP to himself
    - R3 sets NEXT_HOP to 3.3.3.3
- Forces traffic to come to him so he can load-balance
- Is typically used for multiple eBGP sessions to an AS
    - Be careful when using in Confederations!!
- Need to make R2 prefer the path from R1 to prevent the routing loop
    - Make IGP metric to 1.1.1.1 better than IGP metric to 4.4.4.4

MED Churn



- **MED in a RR or Confederation environment can cause an endless convergence loop**
- **Happens as a result of two things:**

  RRs and Confeds “hide” path information

  MEDs are only compared among like Neighbor ASs

- **Two types of “The Churn”**

The Churn – Type I

- **Network must have multiple paths to a prefix via multiple Neighbor ASs**
- **Network must have a single tier of RRs or Sub ASs to have Type I churn**
- **Type I can be fixed today**

  Network must use "deterministic-med"

  Network must follow the deployment guidelines of the RR and Confed drafts

  Drafts state that "intra cluster/SubAS paths must be preferred over inter cluster/SubAS paths"

  Result is that "intra" IGP metrics must ALWAYS be lower than "inter" IGP metrics

The Churn – Type I

- **Still not a great solution**

    IGP change could trigger The Churn

    Networks are bound to a single tier

    Hands are tied in terms of setting IGP metrics

- **For more details please see:**

    “Endless BGP Convergence Problem” -

    www.cisco.com/warp/public/770/fn12942.html

    Includes information on how to identify MED Churn

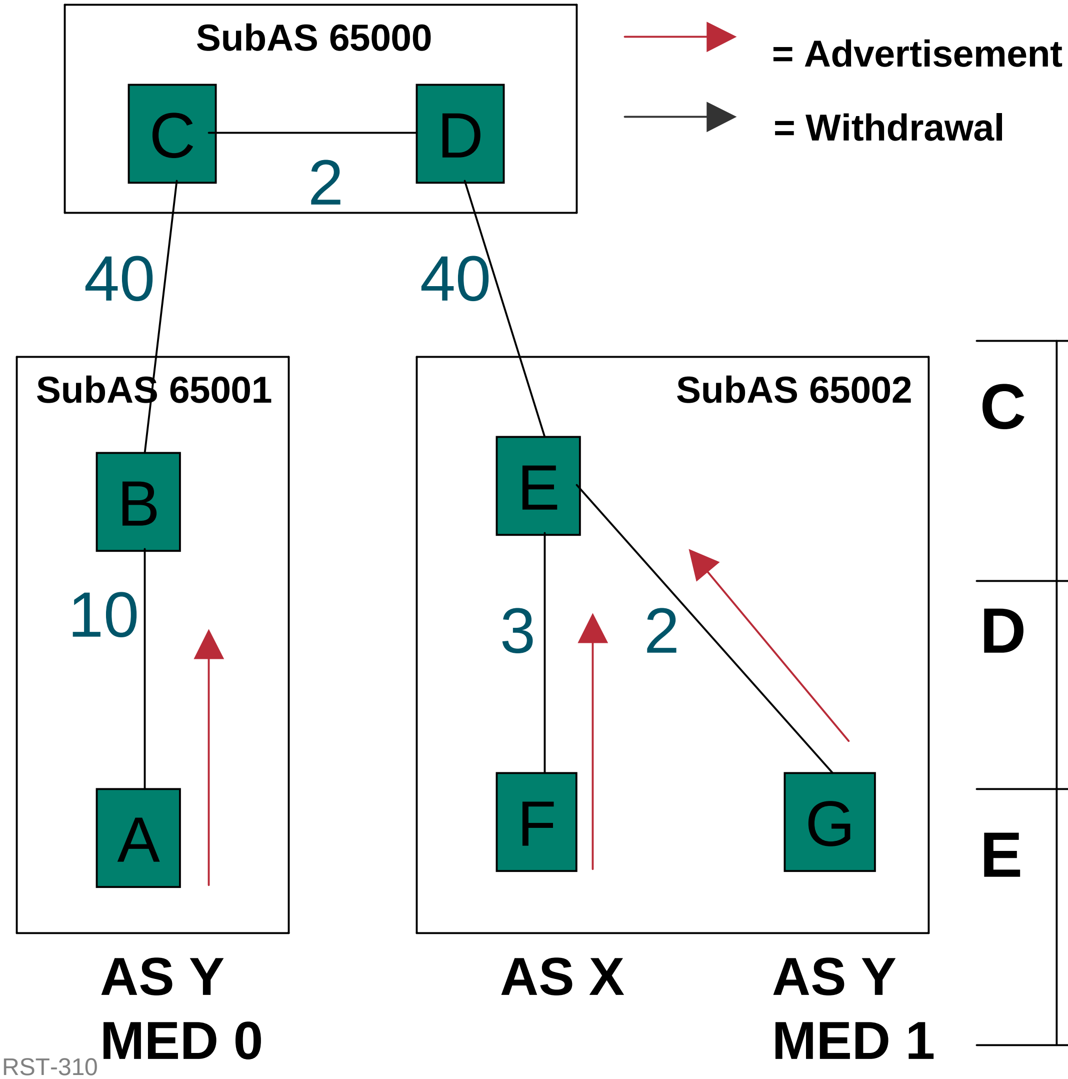
    Includes an example of Type I churn

    Includes information on the solution for Type I

The Churn – Type II

- **Network must have multiple paths to a prefix via multiple Neighbor ASs**
- **Network must have more than one tier of RRs or SubASs**
- **Solution for Type I does not apply**
- **Type II cannot be fixed today with the current decision algorithm**
- **Example …**

The Churn – Type II

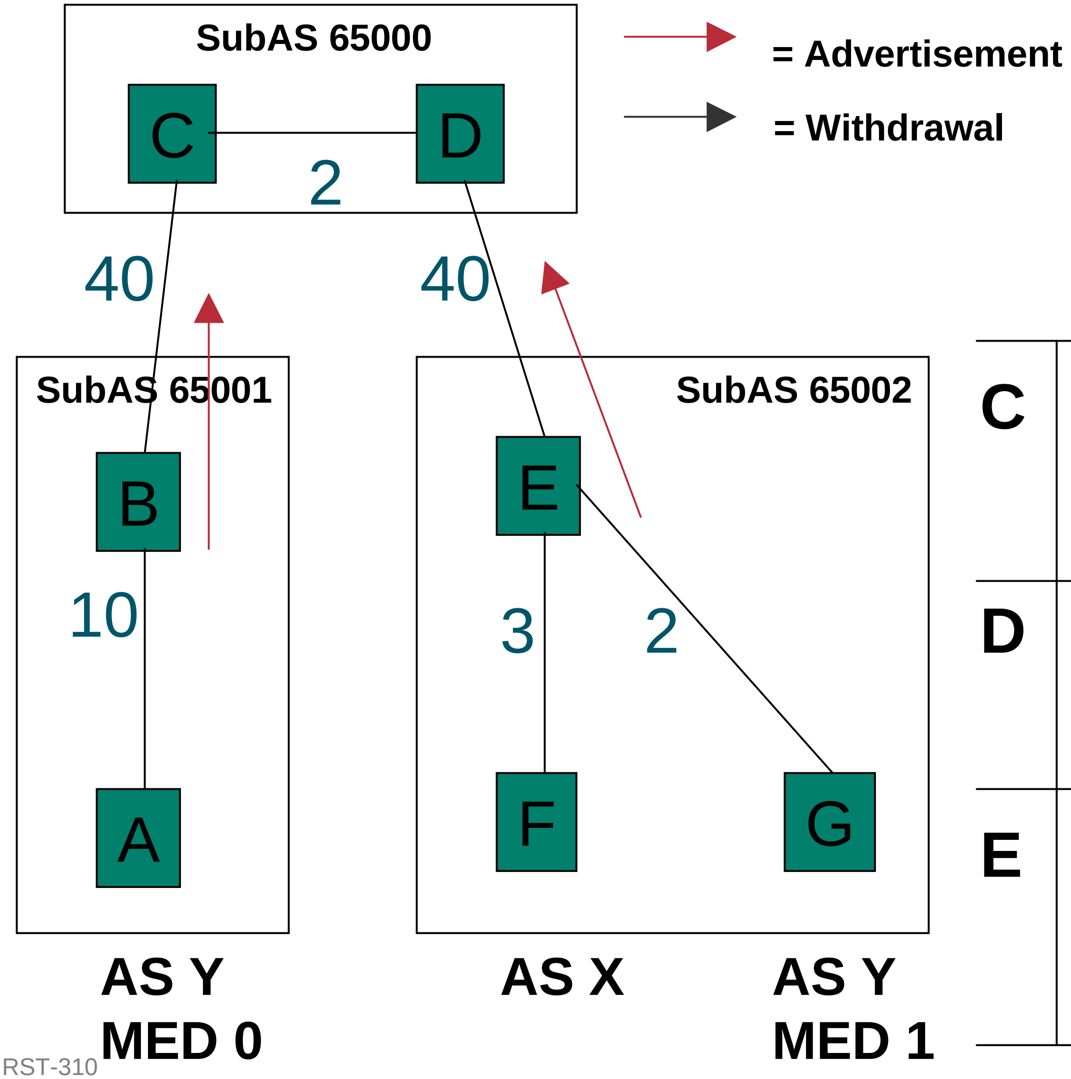


Step 1

– E selects Y1

	AS_PATH	MED	IGP
C			
D			
E	X * Y	 1	3 2

The Churn – Type II

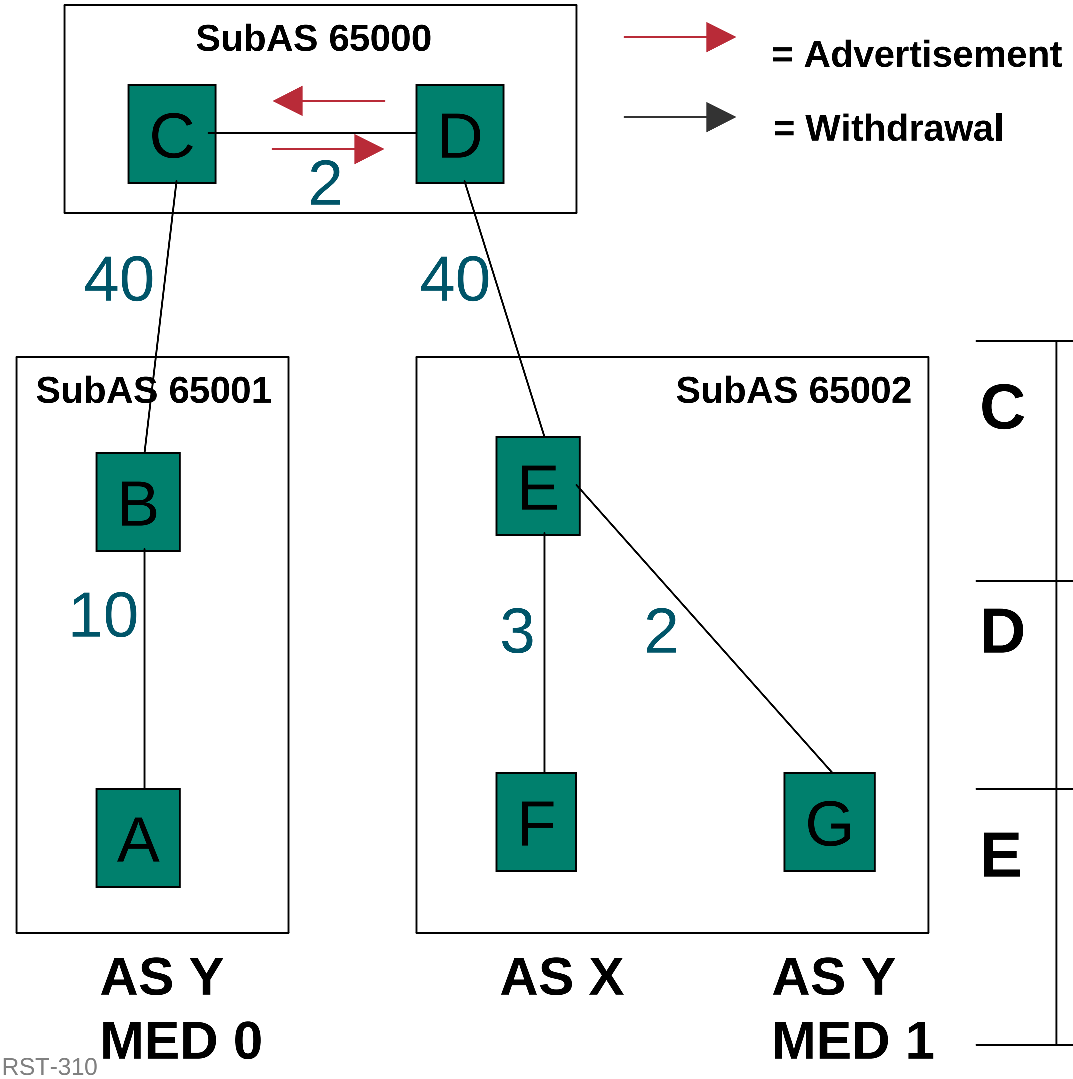


Step 2

- C selects Y0
- D selects Y1

	AS_PATH	MED	IGP
C	* Y	0	50
D	* Y	1	42
E	X * Y	 1	3 2

The Churn – Type II

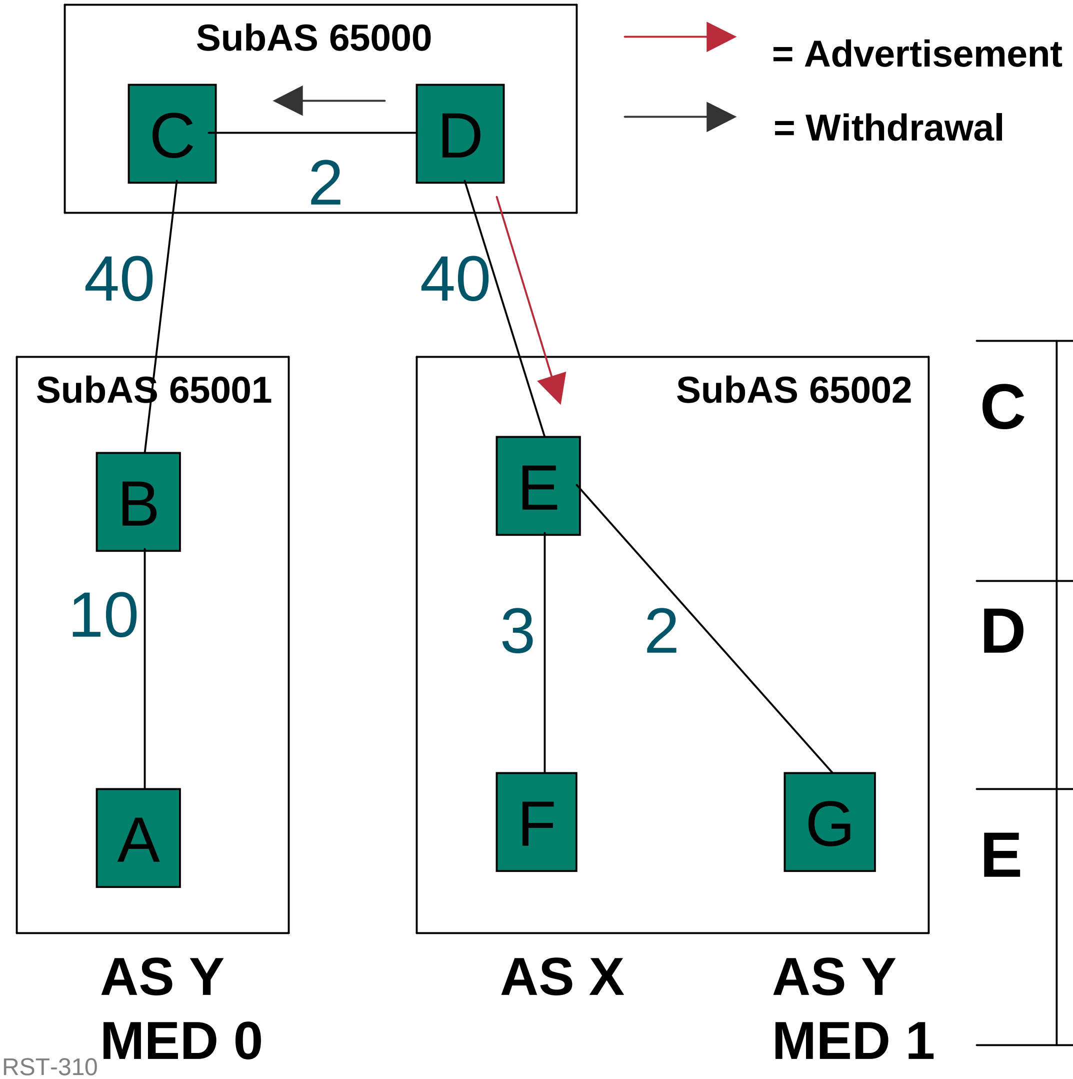


Step 3

– D selects Y0

	AS_PATH	MED	IGP
C	* Y	0	50
	Y	1	44
D	* Y	0	52
	Y	1	42
E	X		3
	* Y	1	2

The Churn – Type II

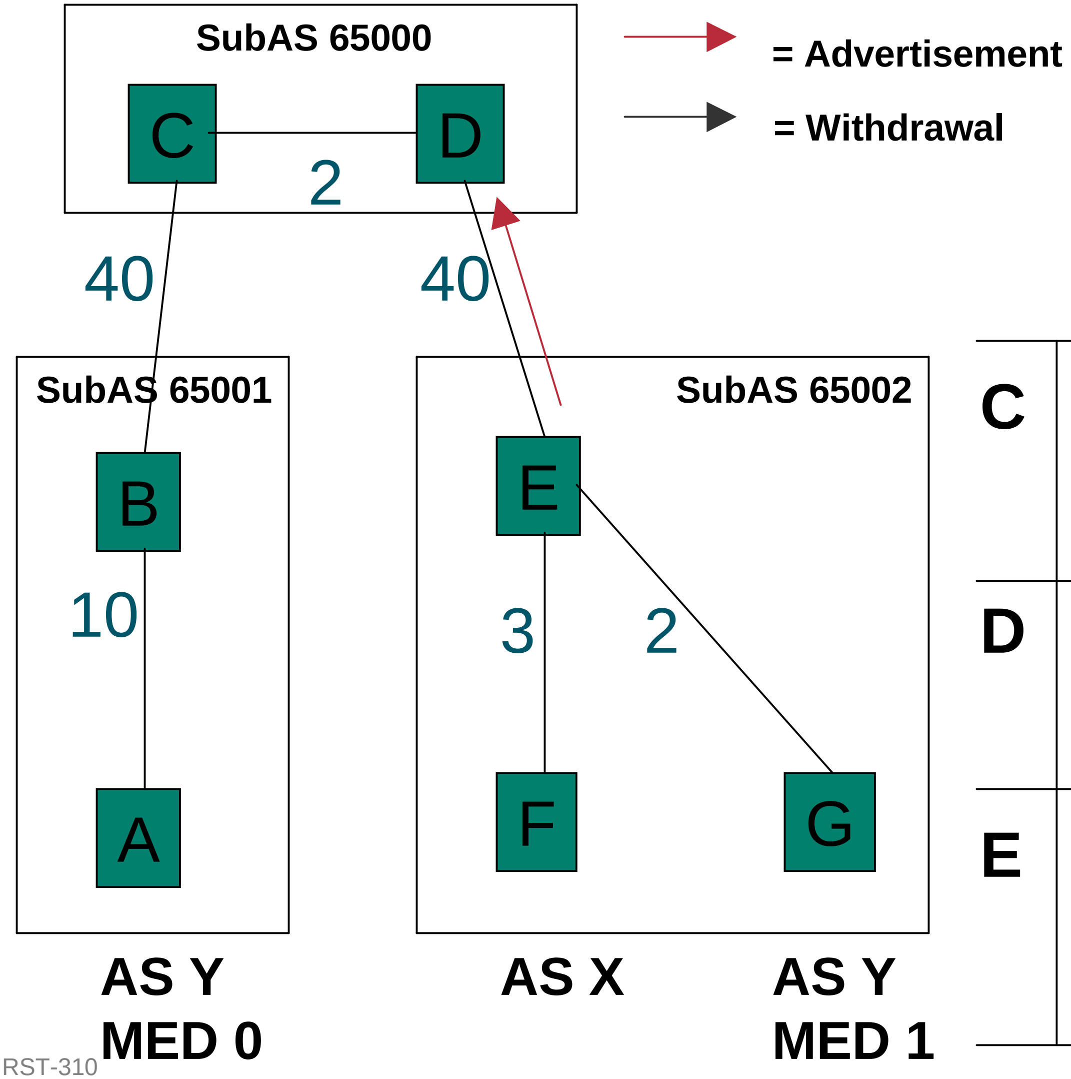


Step 4

– **E selects X**

	AS_PATH	MED	IGP
C	* Y	0	50
D	* Y Y	0 1	52 42
E	* X Y Y	 0 1	3 92 2

The Churn – Type II

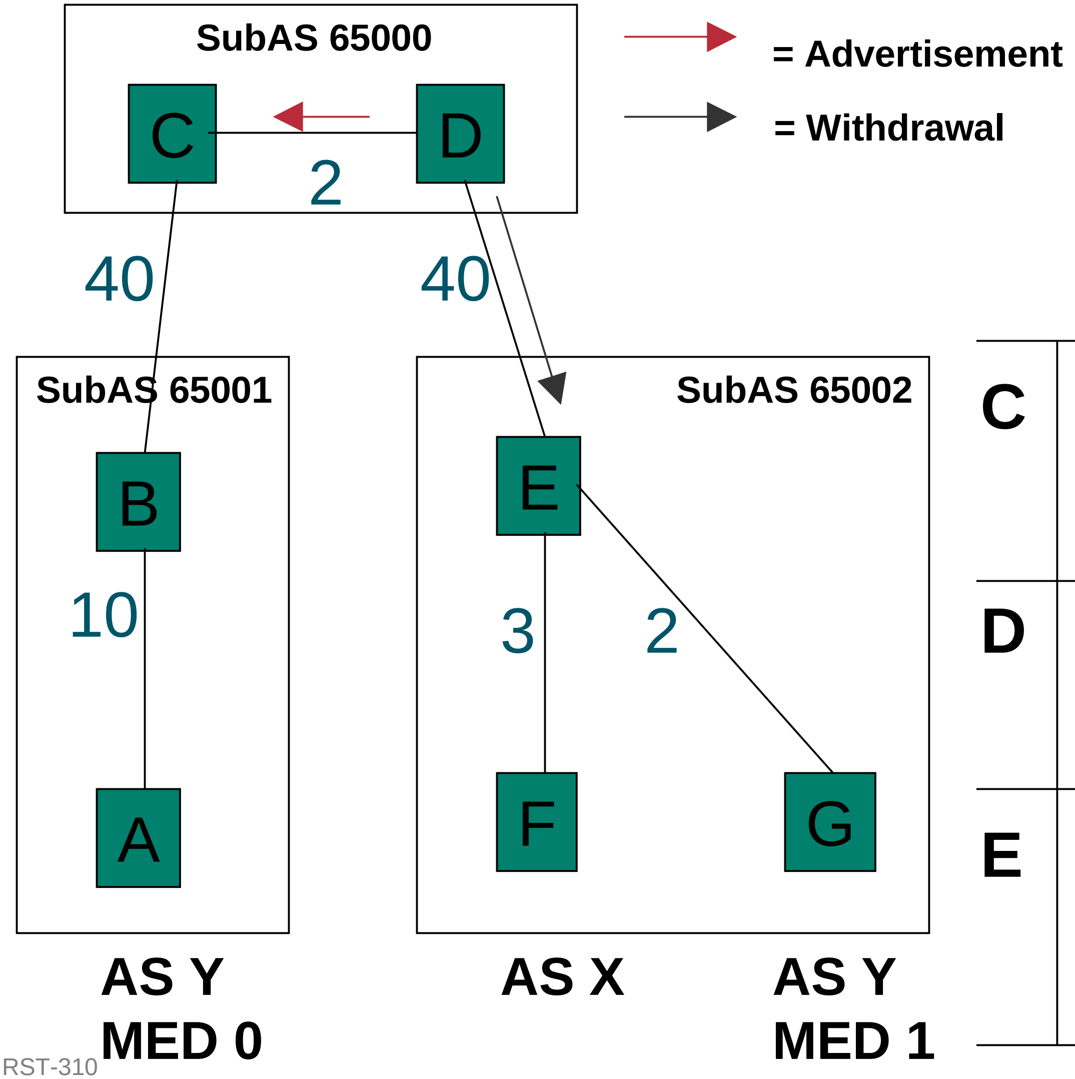


Step 5

– D selects X

	AS_PATH	MED	IGP
C	* Y	0	50
D	Y * X	0	52 43
E	* X Y Y	 0 1	3 92 2

The Churn – Type II



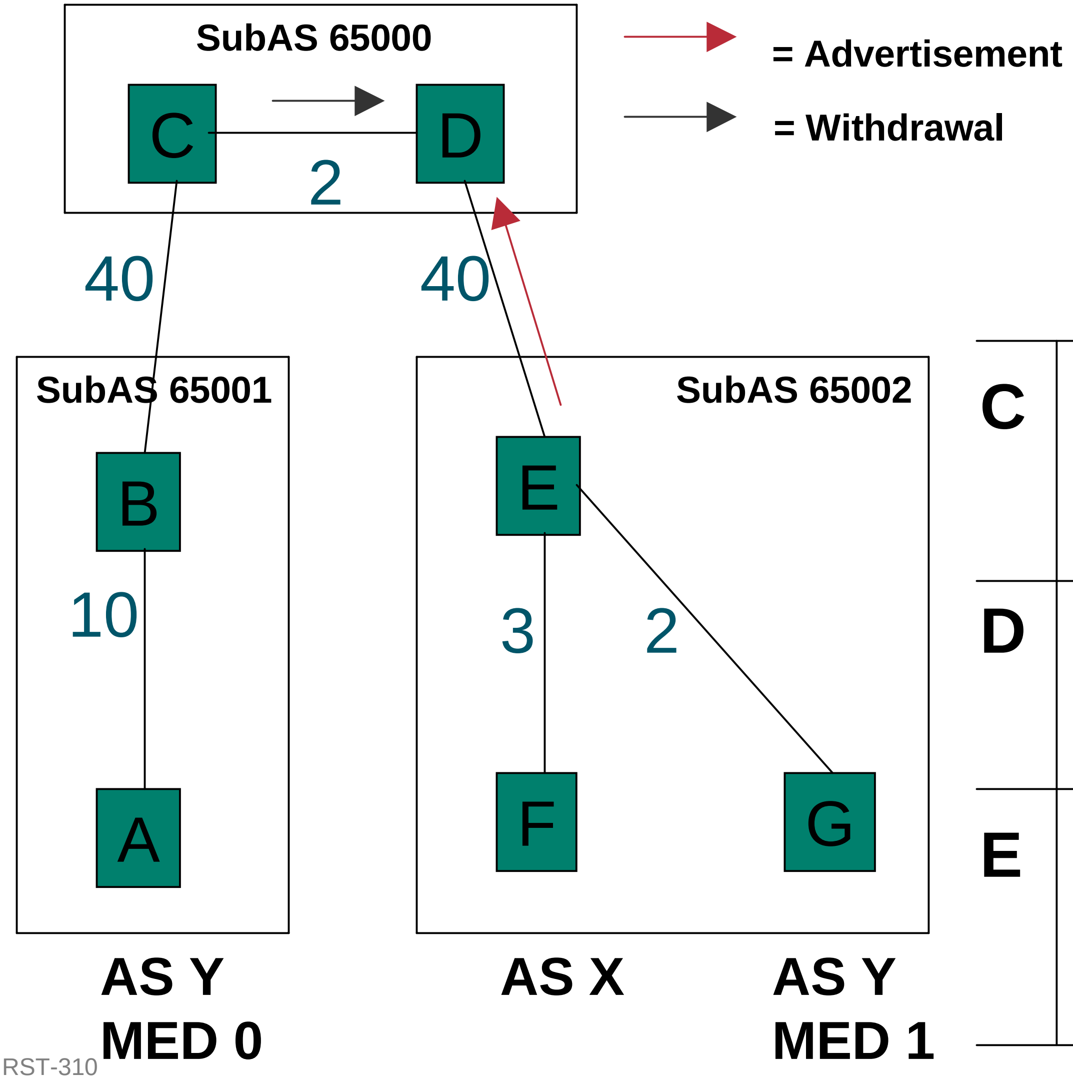
Step 6

– C selects X

– E selects Y1

	AS_PATH	MED	IGP
C	Y	0	50
	* X		45
D	Y	0	52
	* X		43
E	X		3
	* Y	1	2

The Churn – Type II

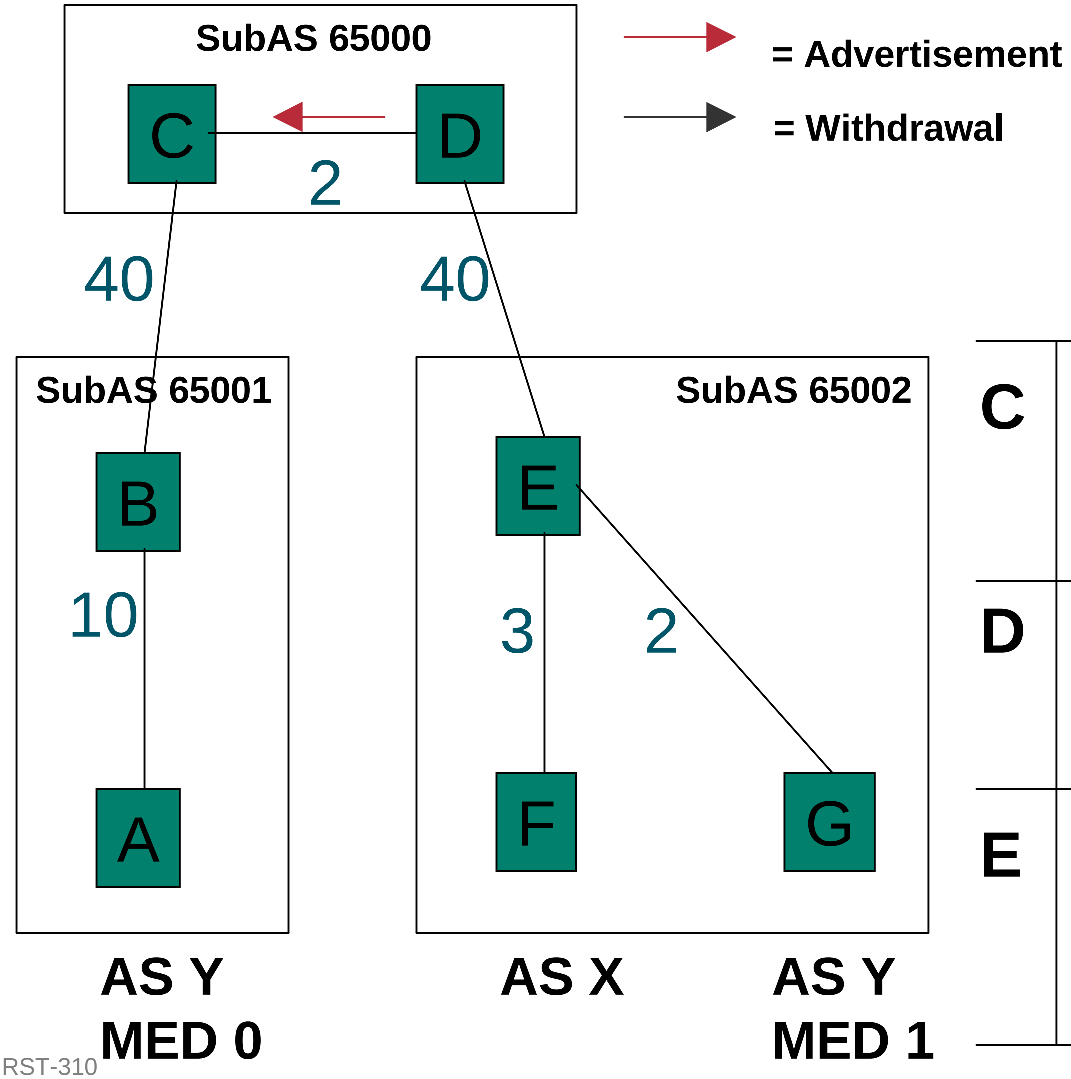


Step 7

– D selects Y1

	AS_PATH	MED	IGP
C	Y	0	50
	* X		45
D	* Y	1	42
E	X		3
	* Y	1	2

The Churn – Type II

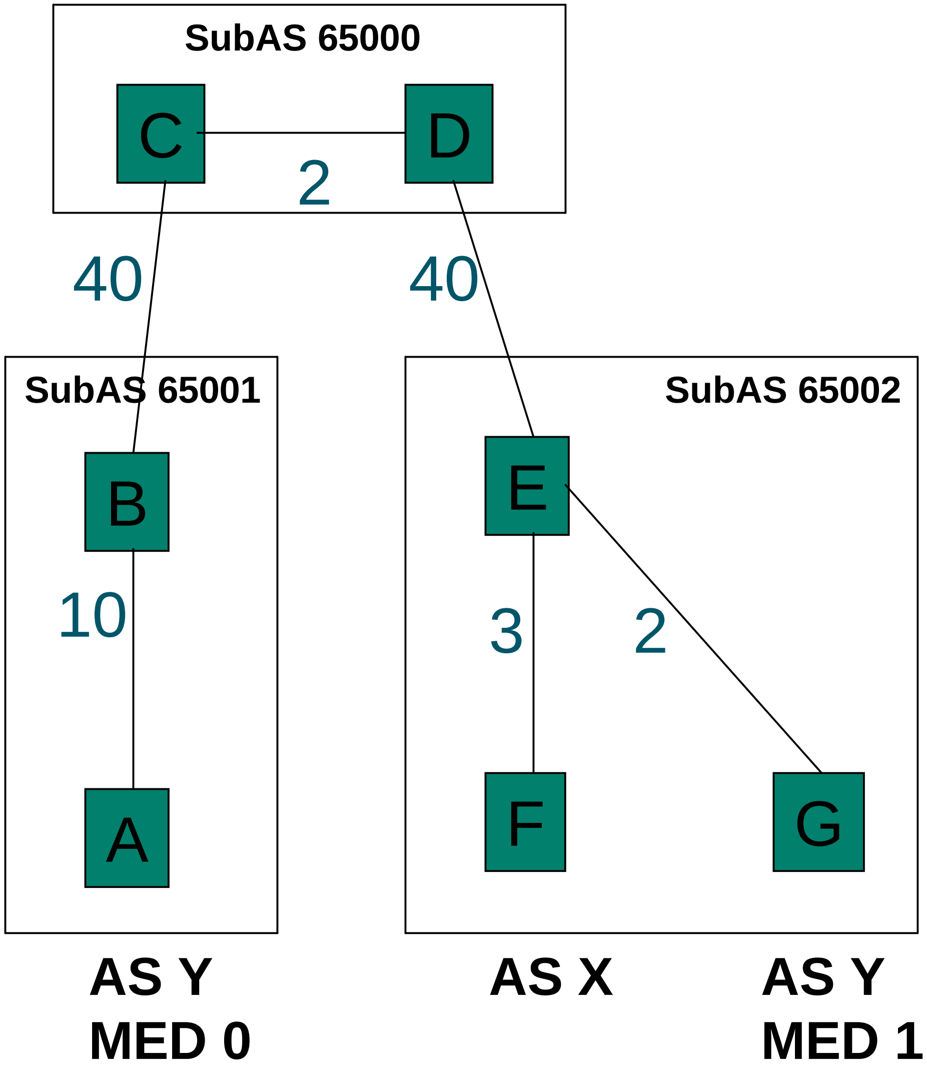


Step 8

- C selects Y0
- This is the same as Step 2
- BGP is in a loop

	AS_PATH	MED	IGP
C	* Y	0	50
	Y	1	44
D	* Y	1	42
E	X		3
	* Y	1	2

The Churn – Type II



- **In a nutshell, the churn happens because E does not always know about the Y0 path but the Y0 path has an effect on what E considers to be his best path**
- **Without Y0, E considers Y1 as best**
- **With Y0, E considers X as best**
- **From C and D's point of view**

  Y0 < Y1 < X < Y0 ← This happens because MED is not compared every time
- **Sequence**

  C selects Y0 and Y0 is propagated to D, E

  E receives Y0 which forces E to select X

  D receives X and selects it over Y0

  C receives X and selects it over Y0

  C sends a withdrawal for Y0

  E receives the withdrawal for Y0 so E now prefers Y1

  C, D receive Y1 but select Y0

  And so on and so on…

Possible Solutions

- **Solution #1 – Make sure E has the Y0 path**

  BGP Peers will need to advertise multiple paths

  BGP will need a new Attribute that will allow a speaker to advertise multiple paths for the same prefix (draft coming soon)

  A BGP speaker will then need to advertise a best path per “Neighbor AS” group IF that path came from an internal peer. This will force C and D to always advertise Y0 to D

- **Solution #2 – Eliminate “Y0 < Y1 < X < Y0” problem**

  Always comparing MEDs accomplishes this

Spotting “The Churn”

- **Two steps to ID the churn in your network**
- **1 – Run “***show ip route bgp | include , 00:00***” once every 60 seconds for ~5 minutes. This will give you a list of routes that have changed within the past minute. If a route is changing every minute then there is a good chance it is churning.**

```
Router#show ip route bgp | include , 00:00
B 2.6.4.0/22 [200/1] via 8.3.4.18, 00:00:32
B 3.8.6.0/23 [200/1] via 7.5.2.5, 00:00:58
Router#
```

Wait 60 seconds…

```
Router#show ip route bgp | include , 00:00
B 17.6.7.0/24 [200/1] via 7.5.2.12, 00:00:17
B 3.8.6.0/23 [200/1] via 7.5.2.5, 00:00:57
Router#
```

3.8.6.0/23 has changed twice in the last 2 minutes. It is possible that this prefix is churning.

Spotting “The Churn”

- **2 – Take a prefix from #1 and do “***show ip bgp x.x.x.x | include best #***“ for a little over 1 minute. If you see a pattern in the best path transition then this prefix is churning. If not, select another prefix from #1 and try again.**

```
Router#show ip bgp 3.8.6.0 | include best #
Paths: (23 available, best #17)
Router#show ip bgp 3.8.6.0 | include best #
Paths: (23 available, best #17)
Router#show ip bgp 3.8.6.0 | include best #
Paths: (23 available, best #17)
Router#show ip bgp 3.8.6.0 | include best #
Paths: (23 available, best #17)
```

Then, the best path changes to #14.

```
Router#show ip bgp 3.8.6.0 | include best #
Paths: (23 available, best #14)
```

Next, the best path changes to #18.

```
Router#show ip bgp 3.8.6.0 | include best #
Paths: (24 available, best #18)
```

Now, the best path is #17 again.

```
Router#show ip bgp 3.8.6.0 | include best #
Paths: (23 available, best #17)
Router#show ip bgp 3.8.6.0 | include best #
Paths: (23 available, best #17)
```

Notice the transition “17->17->14->18->17->17”!!

Repeat Step #2 for another minute just to be sure

Summary

- **Single Tier Networks**

  The churn can be eliminated by using deterministic-med and tweaking your IGP metrics. Another option is to always compare MED.

- **Multi Tier Networks**

  Currently the only “solution” is to always compare MED. A more feasible solution is in the works but it will require BGP to propagate more than one path for a prefix.

Troubleshooting Tips

- High CPU in "Router BGP" is normally a sign of a convergence problem
- Find a prefix that changes every minute

```
show ip route | include , 00:00
```

- Troubleshoot/debug that one prefix

Troubleshooting Tips

- **BGP routing loop?**

  First, check for IGP routing loops to the BGP NEXT_HOPs

- **BGP loops are normally caused by**

  Not following physical topology in RR environment

  Multipath with confederations

  Lack of a full iBGP mesh

- **Get the following from each router in the loop path**

  show ip route x.x.x.x

  show ip bgp x.x.x.x

  show ip route NEXT_HOP

Summary/Tips

- Use ACLs when enabling debug commands
- Enable bgp log-neighbor-changes
- Use bgp deterministic-med
- If the entire table is having problem pick one prefix and troubleshoot it

References

- TAC BGP pages

  http://www.cisco.com/cgi-bin/Support/PSP/psp_view.pl?p=Internetworking:BGP

- BGP Case Studies

  http://www.cisco.com/warp/public/459/bgp-toc.html

- Internet Routing Architectures

  http://www.ciscopress.com/book.cfm?series=1&book=155

- Standards

  RFC 1771, 1997, etc…

  http://www.rfc-editor.org/rfcsearch.html

  http://search.ietf.org/search/brokers/internet-drafts/query.html

Cisco Systems®
Empowering the Internet Generation℠