BGP POLICY AND IMPLEMENTATION

ROUTE-MAP AND BGP ROUTING POLICY

TCOM610
Tawfiq Khan

BGP Policy Implementation

- Cover Principles first
- Then implementation in Cisco IOS syntax
- Configuration syntax varies for different vendors (Cisco, Juniper, Foundry etc.)
- But principles and fundamental policies remain the same
- Jump your reading to Chapter 11 & 12 to get some taste of Cisco IOS syntax

Routing Process

- After BGP session is established, routes are exchanged
- A router can apply Input Policy to manipulate the received routes
  - What routes to accept/reject
  - Record/overwrite route attributes
  - Change/reset route attributes
- A router will also have Output Policy to apply to routes advertised to BGP neighbors
  - Control what route to announce/with what attributes
  - Influence how other networks treat your routes

BGP RIB

- RIB = Routing Information Base
- Adj-RIB-In: BGP RIB associated with each BGP peer; contains all routes from the peer; Before Input Policy is applied
- Adj-RIB-Out: BGP RIB associated with each BGP peer; contains all routes to be advertised to peer; After the Output Policy is applied;
- LOC-RIB: the effective routing table selected as the best BGP routes to be used in local router's routing decision

BGP Policy

- BGP Policy: design input or output policies to apply to different BGP neighbors to influence inbound/outbound traffic behaviors
  - Alleviate congestion
  - Achieve load-balance or load-sharing
  - Achieve redundancy
  - Achieve symmetry routing or least-cost routing
- Input policies: to apply to routes received from peers; affect the outbound traffic behaviors
- Output policies: to apply to routes to be advertised to neighbors; affect the inbound traffic behaviors
- The direction of traffic impacted by routing policies is opposite to each other!

BGP Decision Process

1. Is the Next_Hop available?
2. Largest weight?
3. Largest Local_pref?
4. Locally Originated?
5. Shortest AS_Path?
6. Lowest origin type (IGP < EGP < Incomplete)
7. Lowest MED
8. Prefer EBGP over IBGP route
9. Closest Next Hop by IGP metrics
10. Lowest BGP Router ID/Peer ID

BGP Decision Example 1

```
pop1-tkn#sh ip bgp 12.159.64.0/21
BGP routing table entry for 12.159.64.0/21, version 22816998
Paths: (4 available, best #1)
  Advertised to non peer-group peers:
  210.150.215.177
  22822 18806
    66.185.128.11 (metric 1407) from 66.185.129.80 (66.185.129.80)
      Origin IGP, metric 210, localpref 300, valid, internal, best
      Community: 1668:11531
      Originator: 66.185.128.11, Cluster list: 0.0.0.81, 0.0.1.5
  22822 18806
    66.185.128.11 (metric 1407) from 66.185.129.81 (66.185.129.81)
      Origin IGP, metric 210, localpref 300, valid, internal
      Community: 1668:11531
      Originator: 66.185.128.11, Cluster list: 0.0.0.81, 0.0.1.5
  4725 22822 18806
    143.90.154.145 from 143.90.154.145 (143.90.151.223)
      Origin IGP, localpref 110, valid, external
      Community: 1668:11 1668:30600 1668:32000
```

BGP Decision Example 2

```
pop3-ash#sh ip bgp 12.13.75.0/24
BGP routing table entry for 12.13.75.0/24, version 23917403
Paths: (7 available, best #2)
  Advertised to peer-groups:
    POP_REFLECTOR
  7018 32138
    66.185.128.67 (metric 1007) from 66.185.128.67 (66.185.128.67)
      Origin IGP, metric 0, localpref 100, valid, internal
      Community: 1668:31000
  7132 32138
    66.185.136.2 from 66.185.136.2 (66.10.0.52)
      Origin IGP, localpref 100, valid, external, best
      Community: 1668:31000
1239 7132 32138
    144.223.246.13 from 144.223.246.13 (144.228.241.253)
      Origin IGP, localpref 80, valid, external
      Community: 1668:31000
1239 7132 32138
    144.223.246.129 from 144.223.246.129 (144.228.241.254)
      Origin IGP, localpref 100, valid, external
      Community: 1668:31000
```

BGP Decision Example 3

```
BGP routing table entry for 12.4.97.0/24, version 39835713
Paths: (18 available, best #3)
  Advertised to peer-groups:
    POP_CLIENT
  7018 1239 14452
    66.185.129.83 (metric 1137) from 66.185.129.80 (66.185.129.80)
      Origin IGP, metric 0, localpref 100, valid, internal
      Community: 1668:11 1668:32000
      Originator: 66.185.129.83, Cluster list: 0.0.0.81
  7018 14452
    66.185.129.83 (metric 1137) from 66.185.129.81 (66.185.129.81)
      Origin IGP, metric 0, localpref 100, valid, internal
      Community: 1668:11 1668:32000
      Originator: 66.185.129.83, Cluster list: 0.0.0.81
  1239 14452
    66.185.128.210 (metric 514) from 66.185.128.208 (66.185.128.208)
      Origin IGP, metric 0, localpref 100, valid, internal, best
      Community: 1668:31000
      Originator: 66.185.128.210, Cluster list: 0.0.0.200
  1239 14452
    66.185.128.226 (metric 605) from 66.185.128.225 (66.185.128.225)
      Origin IGP, metric 0, localpref 100, valid, internal
      Community: 1668:31000
      Originator: 66.185.128.226, Cluster list: 0.0.0.230
```

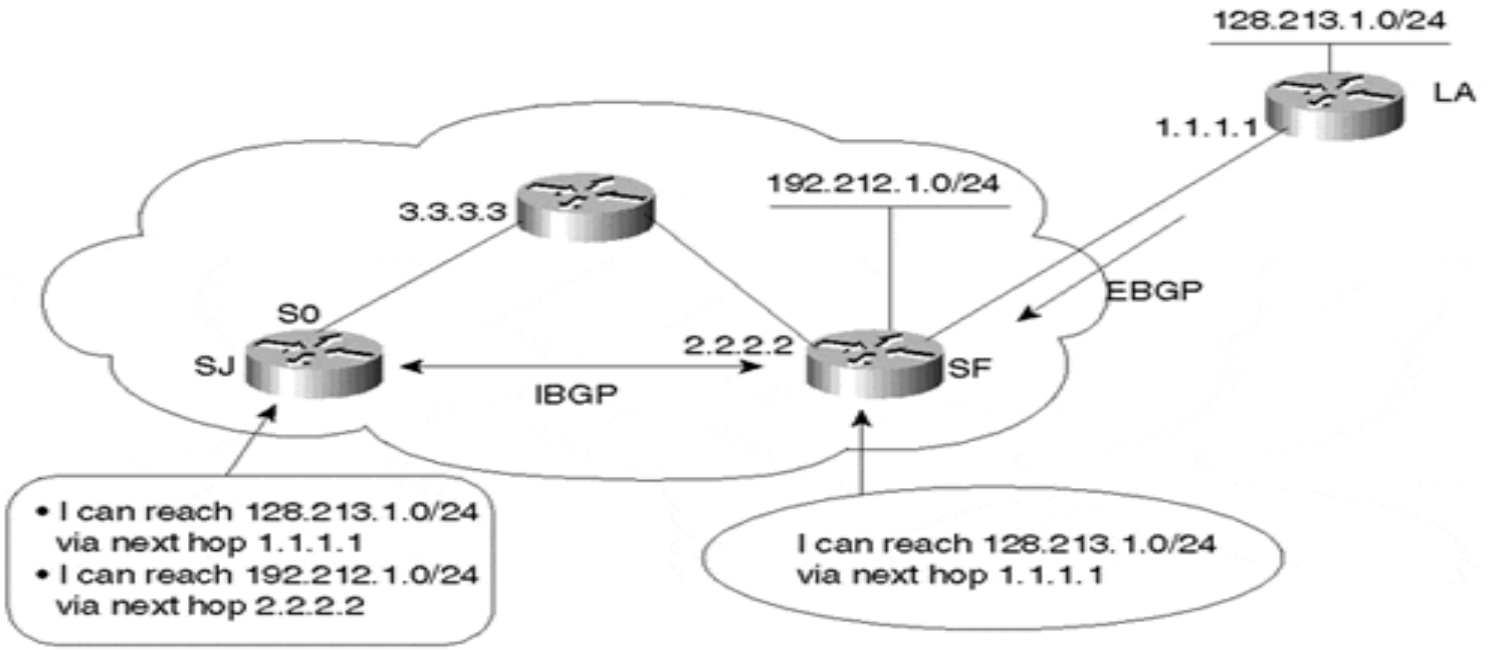
Application of BGP Attributes: next-hop

- Well-known, mandatory, type 3
- EBGP
  - Next_Hop is the IP address of the announcing neighbor
- IBGP
  - Internally originated routes – address of originator
  - EBGP learned – unmodified next hop
  - Or set next-hop-self
- Recursive lookup:
  - Keep going until you find an directly connected interface
  - Necessary because of different methods of expressing next hop information

Next-hop Self

- Remote router's reachability to EBGP learnt routes: through IBGP
- Option one:
  - No next-hop changed
  - advertise a DMZ subnet in your IGP (passive interface)
  - The DMZ address may come from your own IP block or your ISP's address block
  - Then remote next-hop is reachable though IGP
- Option two:
  - Set next-hop-self on IBGP announcement of EBGP learnt routes
  - IBGP peers will forward all traffic to edge peering routers
  - Only requires edge route's loopback address in IGP, no need to bring DMZ into IGP – but you can't traceroute/ping DMZ
  - Alternately, you can also distribute connected networks into BGP so that you can still ping DMZ
  - Adds one extra recursive lookup

Next-hop-Self



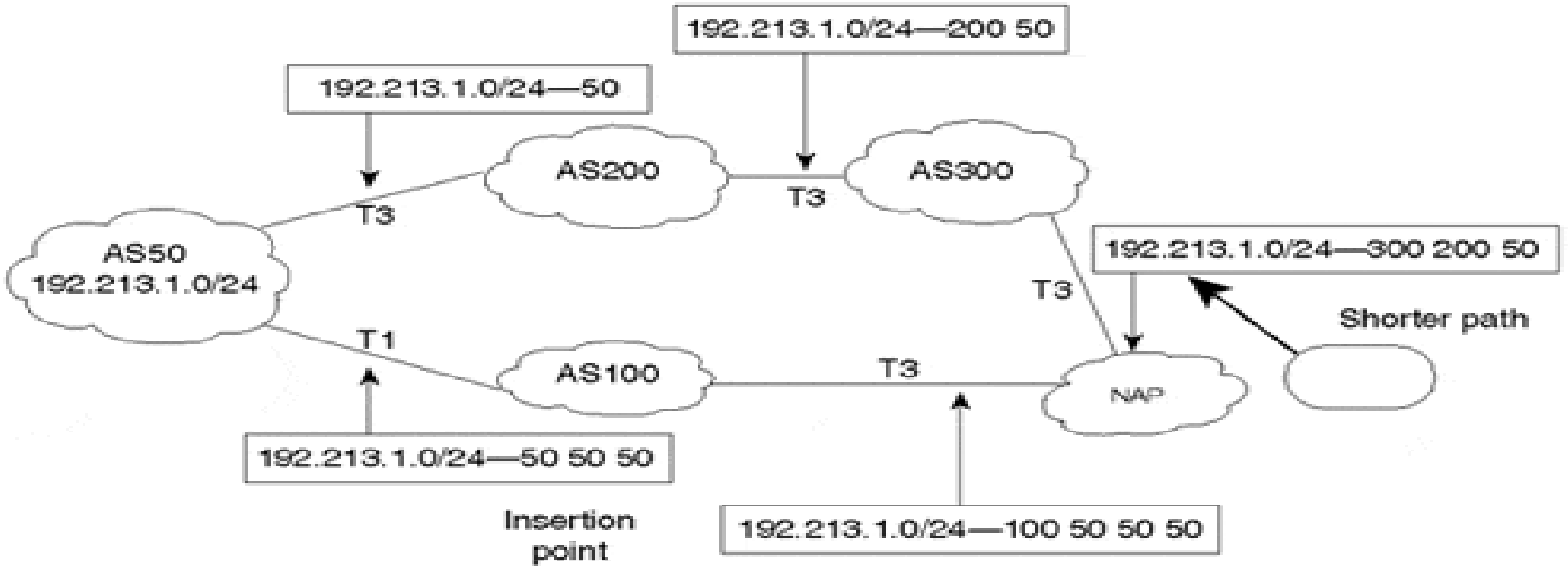
AS-PATH

- Well-known, mandatory, type 2
- Sequence of [sequences | sets] of AS numbers
  - Sequences are ordered, sets are not ordered lists
- Used for loop detection
- Content and length of path used in BGP decision process
- You can selectively prepend your ASN multiple times to the AS_PATH to control inbound traffic flow

AS_PATH Prepending

- Fifth item in the BGP decision process is AS_Path length
- Increase the length by prepending your own AS onto the AS_Path prior to advertising
  - Limit to the number of AS numbers you can prepend: 16 times
  - AS-prepend become ineffective after certain limit, and DOS threat due to memory exhaustion
  - Can only prepend your own AS
- You can not shorten AS_PATH
- Example:
  - AS10 connects to AS100 with T3 and AS200 with T1, and will prefer upstream ISP to send traffic through T3 instead of T1
  - For routes announced to AS200, prepend with AS10 multiple times
  - Other networks see AS10 routes through AS200 with longer AS_PATH

AS_PATH Prepending



Private AS Number

- Private AS pool from 64512 to 65535
  - Must be removed from advertisements to Internet because of not uniqueness
  - Must only be connected to single provider
  - Can configure “remove-private-as” to remove
- Example
  - Single-homed customer with AS65001
  - Runs EBGP with AS1 and announce 172.16.220.0/24
  - AS1 will remove private AS65001 from the route before announcing the route to the Internet
  - Internet sees: 172.16.220.0/24 AS_PATH=1 instead of 1_65001

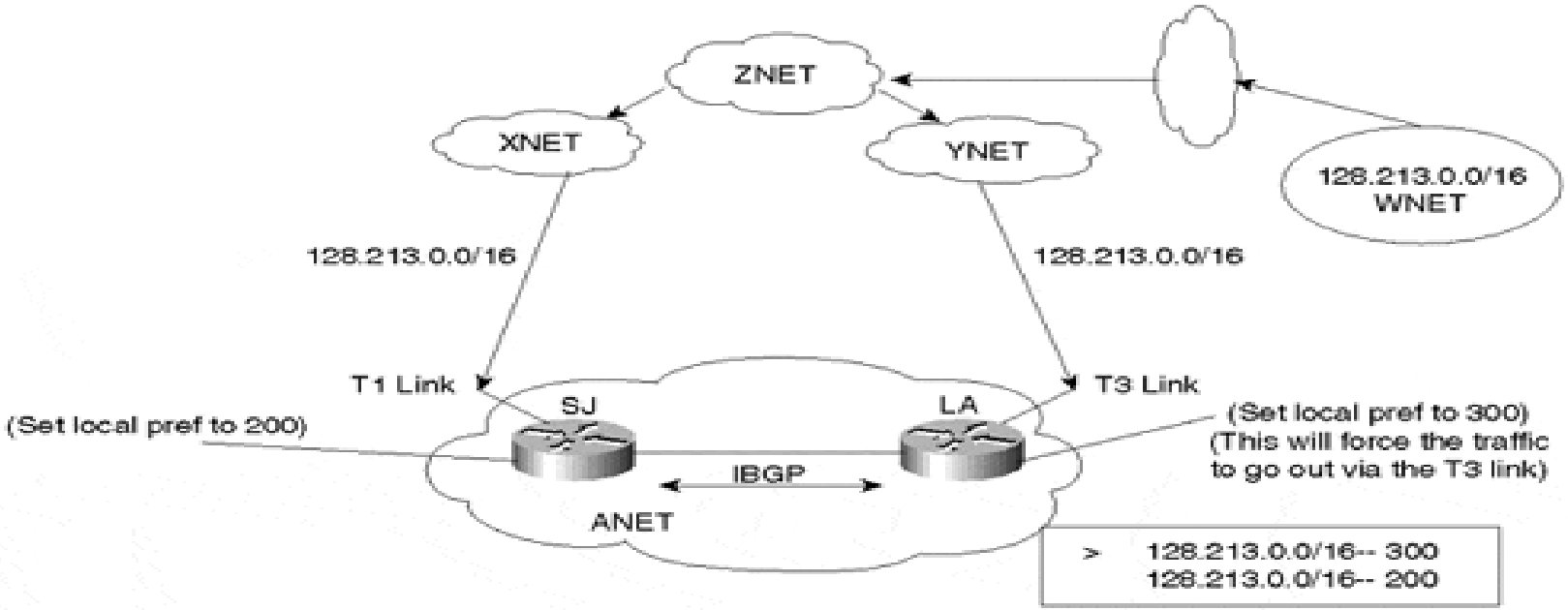
AS-SET

- Aggregates can result in the loss of AS path information if path attributes differ for constituent prefixes
- AS_SET includes ASs that a route has traversed, so that loops can be detected
- Example: inside transitive network AS3
  - Receive 192.213.1.0/24 from AS1
  - Receive 192.213.2.0/24 from AS2
  - AS3 aggregates into: 192.213.0.0/16 with AS_PATH=3_{1 2} and announces the route to the Internet

Local-Pref

- Well-known, discretionary, type 5, non-transitive
- Carry to IBGP peers only
- High priority in decision process so this is an effective way to insure an entire AS exits at the same place for a given prefix under normal conditions
- Influence your outbound traffic exit network
- Default value: 100 – higher is more preferred!!
- Example
  - AS10 connects to AS100 with T3 and AS200 with T1, and will prefer upstream ISP to send traffic through T3 instead of T1
  - AS10 already AS_PATH prepend for routes announced to AS200
  - AS10 can apply higher local-pref value for routes learnt from AS100
  - All inbound/outbound traffic will be through AS100
  - AS200 will simply be used as a backup under AS100 T3 failure

Local_Pref


ZNET
XNET
YNET
128.213.0.0/16
WNET
128.213.0.0/16
128.213.0.0/16
T1 Link
T3 Link
SJ
LA
(Set local pref to 200)
(Set local pref to 300)
(This will force the traffic
to go out via the T3 link)
IBGP
ANET
> 128.213.0.0/16-- 300
128.213.0.0/16-- 200

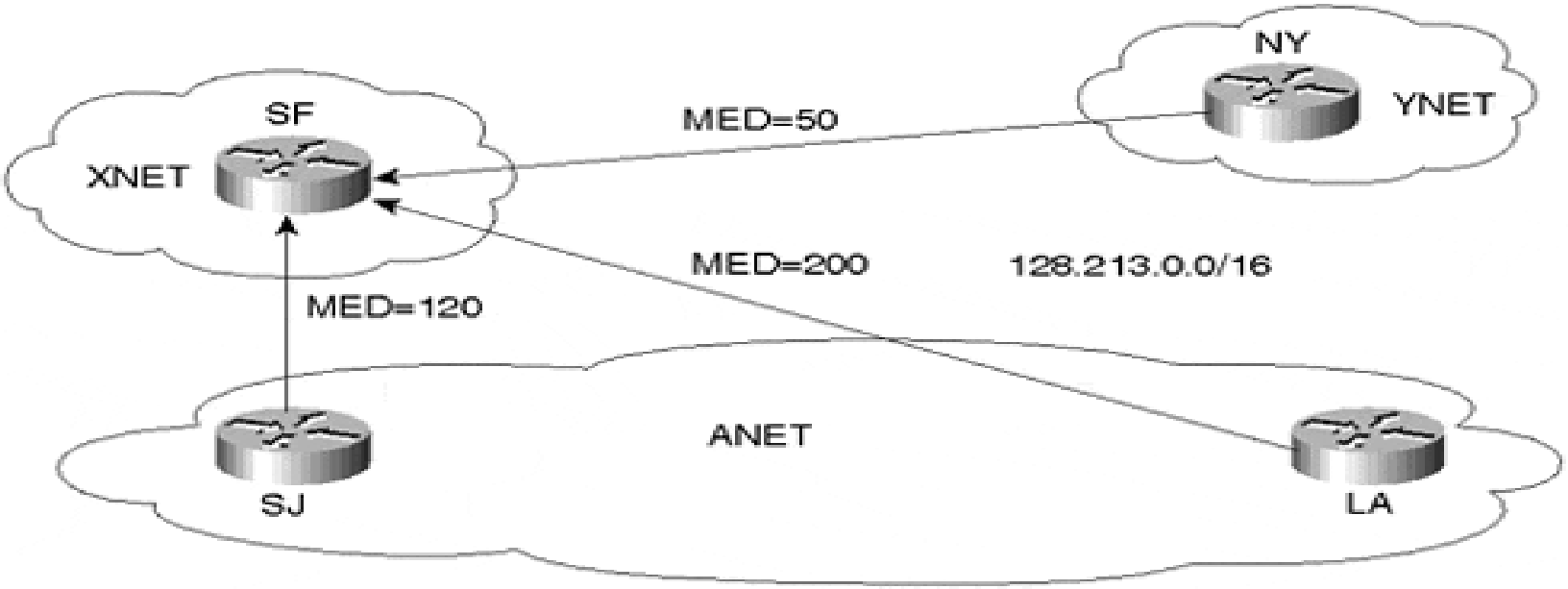
MED

- Optional, non-transitive, type 4
- Used as a hint to external neighbor as to which exit to take your inbound -- their outbound traffic
- Often set to be the IGP metric cost (from route origin to the exit point)
- Default value: 0
  - set metric-type [internal|external|type-1|type-2]
  - Not used to compare routes from different AS
  - Can override with “bgp always-compare-med”
  - Sometime considered not a good way to influence traffic, since it may cause route oscillation problem, especially in RR scenarios

MED Example

- AS10 peers with AS100 in SF and in NY
- SF data center route 128.213.0.0/16 and NY data center route 128.214.0.0/16 are announced to AS100 in both peering sessions
- Without MED, traffic destined to SF may enter into AS10 through NY peering -- AS10 needs to carry traffic all the way from NY to SF
- If for 128.213.0.0/16, AS10 sets MED = 10 in SF session and MED = 200 in NY session and ask AS100 to listen to its MEDs
- Then all traffic destined to SF will enter through SF session, unless SF peering session fails
- Similar behaviors for NY routes if MED is set (200, 10)

MED Example



BGP Community

- Optional, transitive, type 8
- Ways to group a set of routes administratively
- 4 bytes integer value: 0 to 4,294,967,200
- Can be break into (2 byte:2 byte) format, typically set to ASN:value
  - Configure with “ip bgp-community new-format” in Cisco
- Arbitrary unordered set of AS:value pairs
- By default, community is not sent to peers, need to explicitly configure “neighbor x.x.x.x send-community” in Cisco router

BGP Community

- Predefined well-known communities
  - NO_EXPORT – advertised to EBGP peers, but ask them not to export to other networks
  - NO_ADVERTISE – do not advertise this route to any peers
- Use route maps to set the community attribute
  - set community *community-number* [additive]
  - If no additive, then overwrite
- Typical use for BGP community
  - Location of route origin: east, west, europe, asia
  - Type of route: customer, transitive, private, peer, dial
  - Destination-sensitive billing (on-net vs off-net traffic)
  - Special community to influence route attribute manipulations – RFC1998, (http://global.mci.com/uk/customer/bgp/)

Aggregator

- Optional, transitive, type 7
- AS and IP address of router that has generated an aggregate
- Used in conjunction wit Atomic_Aggregate attribute to control information loss due to aggregation
- Cisco use aggregate command to aggregate routes, and associated route-map to set aggregate attributes
- You may decide to announce aggregate only, or aggregate with more specifics

BGP Policy Implementation

- Route-map review
- Route filtering
  - Prefix (NLRI match)
  - As-path
  - Community
- Route filtering process
  - Identify the routes (by attributes match)
  - Permit or deny the routes
  - Manipulate the routes by setting attributes
- Route-policy implementation examples

BGP Policy

- Input policy: apply to routes received from peer
  - Can filter (deny/permit) routes
  - Can select route and apply different policy
  - Influence outbound traffic behavior
- Output policy: apply to routes to be announced to peers
  - Selective route announcement
  - Set route attributes according to your routing policy or your upstream provider policy
  - Influence inbound traffic behavior
- Vendor implementation
  - Cisco: route-map
  - Juniper: policy-statement

Route-map

- Format: route-map *map-name* [permit | deny] *seq#*
  - *map-name* is name you wish to associate with route map
  - *seq#* is any number from 1 - 65535
- Example
  - route-map joe_net permit 10 (blah)
  - route-map joe_net permit 20 (blah) ...

Route-map

- Process the map according to sequence number
- If match then
  - if permit, apply “sets” and use route
  - if deny, drop route
  - Any action will break out of list
- Else process next in sequence
- Implicitly deny at end of map

Route-map Match

- as-path
- community
- clns
- interface
- ip
  - address
  - Next-hop
  - route-source
- route-type
- tag
- length
- others?
- metric

Route-map Set

- as-path
- community
- ip
  - next-hop
  - default next-hop
  - precedence
- tos
- level
- local-pref
- metric
- metric-type
- next-hop
- origin
- tag
- weight
- clns, dampening,...

Route Filtering

- Need to match prefix and the mask
- Cisco has several different ways to filter
  - ACL
  - Extended ACL
  - Prefix-list: most flexible
- Can be used for route filtering (control plane) as well as packet filtering (data plane)
- Typically with multiple clauses: a route/packet goes though the filter -- leave the filter if matched or denied
- With an implicit deny at the end of the filter

Filtering

- Filter both route announcements and protocols
  - Matching prefixes
  - Matching prefix/mask pairs
  - Matching AS numbers
  - Matching community strings
- Filters can be applied at various boundaries
  - BGP peering (EBGP and IBGP)
  - Protocol boundaries between BGP and other protocols (IGP) such as redistribution

Prefix List

- Several advantages
  - Significant performance improvement in loading and route lookup of large lists
  - Support for incremental updates
  - More user-friendly command line interface
- Command format
  - [no] ip prefix-list *list-name* [seq *seq-value*] deny | permit *prefix/length* [ge *ge-value*] [le *le-value*]

Prefix list

- Attributes
  - *list-name* – a string identifier
  - seq *seq-value* – optional - specify the sequence number
  - defaults to five plus current maximum
  - deny or permit – action to take
  - *prefix/lengths* – normal notation
  - ge *ge-value* – match lengths greater than or equal to the given value
  - le *le-value* – match lengths less than or equal to the give value

Prefix List Example

- Exact matches
  - ip prefix-list aaa deny 0.0.0.0/0
  - ip prefix-list aaa permit 35.0.0.0/8
- In 192/8, accept up to /24
  - ip prefix-list aaa permit 192.0.0.0/8 le 24
- In 192/8, deny /25+
  - ip prefix-list aaa deny 192.0.0.0/8 ge 25
- In all address space, permit /8 - /24
  - ip prefix- list aaa permit 0.0.0.0/0 ge 8 le 24

Example

- In all address space, deny /25+
  - ip prefix-list aaa deny 0.0.0.0/0 ge 25
- In 10/8, deny all
  - ip prefix-list aaa deny 10.0.0.0/8 le 32
- In 204.70.1/24, deny /25+
  - ip prefix-list aaa deny 204.70.1.0/24 ge 25
- Permit all
  - ip prefix-list aaa permit 0.0.0.0/0 le 32

AS_PATH Matching

- Treat the AS_path as a string
- Numbers, space, braces, parentheses, comma
- Normal regexp syntax with single addition
  - The underscore (_) matches a comma, left brace, right brace, right parenthesis, left parenthesis, beginning of string, end of string or a space – basically everything but a number
  - ^ for beginning and $ for end

AS_PATH Regular Expression

- Examples
  - Locally originated routes ^$
  - All routes .*
  - Routes originated by AS1239: _1239$
  - Routes going through AS1239: _1239_
  - Routes learned from AS1239: ^1239_
- Don’t forget the underscore, since 12391 is also a valid AS_num

Creating AS_PATH list

- At the configuration prompt in Cisco IOS
  - ip as-path access-list *num* permit *regexp*
- Creates access list for AS_Path information

Applying as-path filter

- In BGP neighbor session use the “filter-list” keyword
  - neighbor *addr* filter-list *num* [in | out]
- In route map use the “as-path” keyword
  - match as-path *num*
- Apply the route-map in the BGP process neighbor configuration
  - neighbor *addr* route-map *num* [in | out]

AS_PATH application

- Often use an AS_Path prepend to force a particular path to be preferred

```
router bgp 65001
neighbor 129.237.125.185 route-map uglymap out

route-map uglymap
set as-path prepend 65001 65001 65001
```

Community String Matching

- Format of list: ip community-list *num* permit [*val*]
  - *num* is a community list number
  - *val* is a community value such as 1239:1002
- Match on information in route map
  - match community *num* [exact]
- A peer must have the corresponding configuration
  - Set community string
  - Have “send-community” configured on peering session

Community String Filtering

- A community list is a group of communities that can be used in match clause of a route map
  - ip community-list *community-list-number* {permit | deny} *community-number*
- Examples
  - ip community-list 10 permit 1:234
  - ip community-list 10 permit 1:456
  - ip community-list 10 deny 1:100
  - ip community-list 10 permit 1:250

BGP Community String Filtering

- Using the keyword exact in the route-map match statement says the community has to be an exact match, which changes the behavior of the community list
- Example
  - route-map simple permit 10
  - match community 10 exact-match
  - set weight 255
  - set community 123:123 additive
  - route-map simple permit 20
  - set weight 100
- Example community lists
  - ip community- list 10 permit 1:234
  - ip community- list 10 permit 1:456
  - ip community- list 10 deny 1:100
  - ip community- list 10 permit 1:250

Community String Example

- Example communities to apply to the lists 1:234 1:100 1:234, 1:456 1:250, 1:234 1:100 1:250 1:100, 1:234 1:100
- Example - change the route map
  - route-map simple permit 10
  - match community 10
  - set weight 255
  - set community 123:123 additive
  - route-map simple permit 20
  - set weight 100
- Now apply example communities

Common Route-maps

- Match on prefix (prefix-list), as-path (filter-list), community
  - Selectively permit/deny subset of routes based on route atributes
- BGP Attribute Setting
  - Local-preference
  - AS-path prepending
  - Community value
  - MED

BGP Policy Implementation

- Peer Group
- Route Aggregation

BGP Peer Group

- Group of BGP neighbors that share same update policies
- Instead of defining same policies for each neighbor, define a group name and shared policies
- Results in reduction of repetitive processing at router as well as administrative savings on configuration

Peer Group Example

- RTA has an internal peer group which implements (common) internal policies for routers inside AS (IBGP peers)
- RTA has an external peer group which implements (common) external policies: default only, default plus customer routes, full internet routes etc.
- Individual peer may customize its inbound policy

Peer-group Example

```
router bgp 1
neighbor INTERNALMAP peer-group
neighbor INTERNALMAP remote-as 1
neighbor INTERNALMAP route-map INTERNAL out
neighbor INTERNALMAP filter-list 1 out
neighbor INTERNALMAP filter-list 2 in
neighbor 172.16.11.1 peer-group INTERNALMAP
neighbor 172.16.13.1 peer-group INTERNALMAP
neighbor 172.16.12.1 peer-group INTERNALMAP
neighbor 172.16.12.1 filter-list 3 in
```

BGP Aggregation

- CIDR and Supernetting
  - not supported in BGP2 and BGP3
  - key feature of BGP4
- Applied to BGP routing table
  - as opposed to network command, which verify with routes in IP routing table
  - can be performed if at least one specific route of aggregate exists in BGP table

Aggregate only, suppressing specifics

- Aggregate is advertised, all specific routes of that aggregate are suppressed
  - assumes no additional benefits from advertising specific routes
- Example
  - AS100 owns 172.16.0.0/24 to 172.16.15.0/24, aggregate into 172.16.0.0/20 and advertise this aggregate only to its provider

Aggregate plus Specific

- Aggregate is advertised, some specific routes of that aggregate are also advertised
  - A customer is multi-homed to a single provider
  - Aggregate only: traffic will be over only one link
  - If aggregate plus all specifics with MED will draw inbound traffic into the closest locations and also achieve redundancy under failure
  - May also want to advertise specifics with NO_EXPORT community so that upstream provide will not leak out specifics to Internet
  - May also consider aggregate with subset of more specific

Example

- Example
  - AS100 owns 172.16.0.0/24 to 172.16.15.0/24
  - Multi-homed at NY and SF
  - Options
    - AS100 can advertise aggregate to AS200, but no load balancing
    - AS100 can advertise specifics to AS200, allows load balancing, at cost of more complex tables
  - AS200 advertises only 172.16.0.0/20 outbound
- Providers can block more specific routes through peering agreements
  - can have customers tag specific routes with community attribute

Example

- May want to advertise only a subset of specific routes to provider
- Example
  - AS100 multihomed at NY and SF
  - prefer that networks near SF be accessed via SF link, networks near NY be accessed via NY link
- Example
  - approach on SF link, advertise aggregate and SF networks on SF link, advertise aggregate and NY networks in NY link
  - AS200 uses specific routes where they exist, aggregate in other cases (not SF or NY, for example)
  - AS200 can use NO-EXPORT to send aggregate only outbound

Cisco Commands

- Router bgp 1
- Aggregate-address 172.16.0.0 255.255.0.0 [summary-only | suppress-map MAP | as-set advertise-map MAP]

- Next-hop = 0.0.0.0 for aggregate
- More specifics will be in suppressed state

AS-SET

- Attributes from specific routes will be lost
- AS-SET contains set of all elements contained in all paths being summarized
  - AS_Path, community attributes
- Can cause instabilities due to changes in individual elements of set of attributes
  - increases withdraw and update frequency

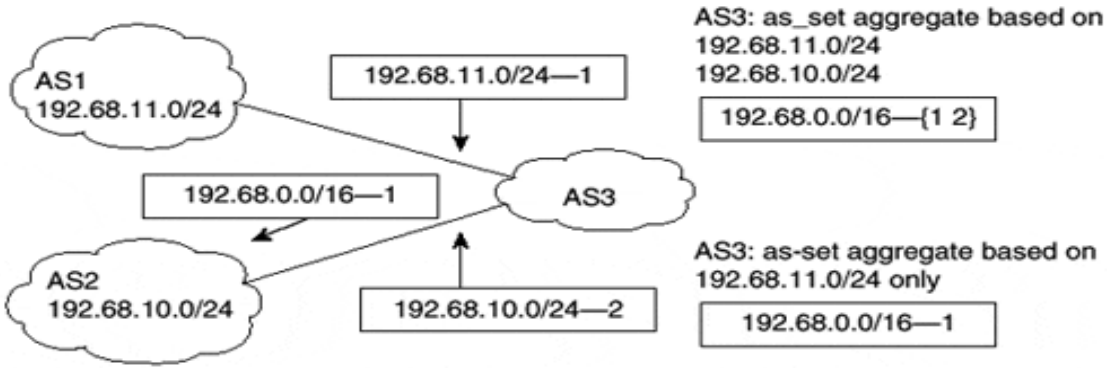
Changing Attributes of aggregates

- Might have undesirable attributes in aggregate
  - might inherit into AS-SET from individual routes
- Examples
  - NO-EXPORT from specific route might prevent aggregate advertisement
  - preferences for particular aggregates
- Multihomed customer advertising aggregate on different links
- Customer wants to use different MEDs to influence inbound traffic to their AS

Aggregate from Subset

- Aggregate may summarize routes from different ASs
- Want to specify which routes are included in aggregate
- Example usage
  - hub and spoke, leaf ASs contain subset of aggregate
  - hub AS excludes more specific routes from leaf ASs
  - hub's advertisement of aggregate to leaf excludes AS number of that leaf to prevent discard

Example



- AS3 receiving routes from AS1 and AS2
- aggregate AS-SET would have AS_Path of {1 2}
- by default, aggregate sent to leaf ASs would be discarded
- need to exclude appropriate AS number for the spoke to accept the aggregate