Implementing Policy-Based Routing

Policy-Based Routing

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
Policy-Based Routing on 8212-48FH-M, 8711-32FH-M, 88-LC1-52Y8H-EM, and 88-LC1-12TH24FH-E	Release 24.3.1	Policy-Based Routing is now supported on these fixed systems and line cards: • 8212-48FH-M • 8711-32FH-M • 88-LC1-52Y8H-EM • 88-LC1-12TH24FH-E
Policy-Based Routing on 88-LC1-36EH	Release 24.2.11	You can now create customised routing policies based on different parameters such as IP address, port numbers, or protocols. With Policy-Based Routing (PBR), you can enhance your network security by steering sensitive data away from potentially vulnerable network segments. Also, by allowing you to distribute traffic across multiple paths, PBR can help prevent traffic congestion in your network. This feature is supported only on 88-LC1-36EH.

Policy-Based Routing (PBR) gives you a flexible means of routing packets by allowing you to configure a defined policy for traffic flows, reducing reliability on routes derived from routing protocols. Moreover, PBR allows you to prioritize and provide a specific routing path for certain types of traffic (for example, VoIP, video conferencing) based on the service-level agreement (SLA).

Unlike traditional routing which is based on destination IP address alone, PBR allows you to route packets based on different parameters such as IP address, port numbers, or protocols. For example, you can implement

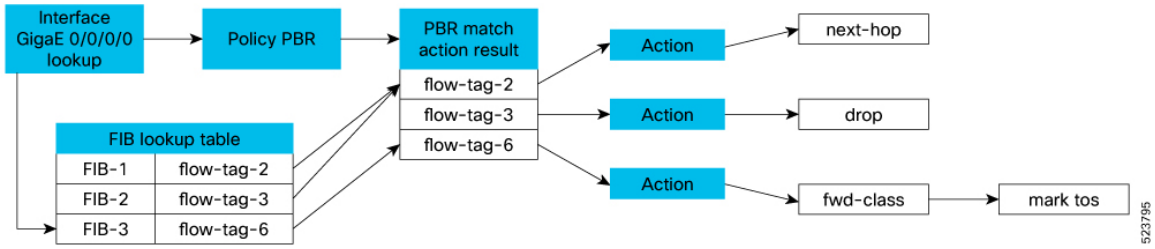
routing policies to allow or deny paths based on the identity of a particular end system, an application protocol, or the size of packets.

Flow-tag Support

A provider edge (PE) router directs traffic toward the core through the routes learnt. It also installs policies on your interfaces to offer value-added services based on your profiles. The administrators must manually scan route-tables and install traffic-path selection as Access Control Lists (ACLs) on your interfaces for proper traffic forwarding. However, with the scale of route prefix-lists learned from the core, the hardware resource requirements to program policies grows high. When route updates change, it requires administrators to manually update these policies. Moreover, these route policies are applied per prefix list and do not allow per interface selection to associate these policies on a per customer basis, which can be a challenge.

To simplify policy management and improve scalability, you can classify the traffic early in the routing domain as routes are learned, and mark them with metadata or a tag in the Forwarding Information Base (FIB). Then, forwarding-path rules can be defined against this flow-tag value. This could reduce the need for manual updates and make the system more scalable and efficient.

The following figure illustrates the process:



Flow-tag is set and distributed via the Routing Information Base (RIB) as a policy attribute of the Forwarding Information Base (FIB) entry in the FIB lookup table. Route Policy Language (RPL) is used to create these flow tags through the "set" operation. The flow tag is then referred to in the PBR policy, where it is associated with specific actions or policy rules against the flow tag's value.

Forward-Class Support

The PBR class-map defines the matching criteria for classifying a particular type of traffic, while the forward-class defines the forwarding path these packets must take. Once a class-map is associated with a forwarding-class in the policy map, all the packets that match the class-map are forwarded as defined in the policy-map.

Traffic Engineering (TE) tunnels are then used to direct traffic along specific paths through the network. These TE tunnels can be associated with a forward-class which determines the forwarding path for the packets.

The use of the **auto-route** command allows the TE interface to be exported to the routing protocol module, associating the route in the Forwarding Information Base (FIB) database with these tunnels.

The system can support up to eight forward-classes with eight TE tunnels each, allowing a maximum of 32 TE tunnels to be associated with the destination route.

Access-Group Support

PBR supports matching on an ACL group ID. You can specify large prefix lists and long lists of specific permit/deny lists. The prefix lists serve as a filter for routing updates, and the permit or deny lists determine what action to take when a match is found.

Permit/Deny Actions in ACL Groups

An access control entry (ACE) can be either a HIT or a MISS. When the ACE is a HIT, the following actions are performed:

- Permit - Stop processing ACL, and if
  - the type of class is “match-any”, proceed to take policy-map action
  - the type of class is “match-all”, proceed to the next ACL or “match” statement in the class-map considering that you already have a match (true) for an existing ACL.
- Deny - Stop processing ACL and if
  - the type of class is “match-any”, proceed to the next ACL (or “match”) statement in the class-map if there is no match for the existing ACL.
  - the type of class is “match-all”, exit the class. No policy action is taken, and the next class is processed as per the specified order.

When the ACE is a MISS (didn’t match any permit or deny statement), the following actions are performed:

- Proceed to the next "match" statement.
- If the “match” statement does not provide a match, then skip that class. No policy action is taken for that class, and proceed to check the packet against the next class in order, or proceed to L3 forwarding lookup.

Note An explicit "deny" entry is not supported in ACL groups that are embedded in class-maps.

Supported Match and Set Operations

The following table illustrates the match/set criteria that is supported by PBR:

Table 2: Supported Match and Set Operations

Criteria	match/set
source ip	match
destination ip	match
source protocol/port	match
destination protocol/port	match
nexthop ip	set
nexthop vrf	set
nexthop ip+vrf	set
forward-class	set
access-group	match

Criteria	match/set
flow-tag	match
ip protocol	match
tcp-flag	match
port-range	match

Restrictions for Implementing Policy-Based Routing

The following are the restrictions for implementing Policy-based routing:

- QoS Group and Flow-tag are not supported together at the same time.
- PBR is not supported on Bridge Group Virtual Interface (BVI) and Pseudowire Headend (PHWE) subinterfaces.
- BGP Flowspec feature and PBR are not supported together on the same interface.
- A route-policy can have either 'set qos-group' or 'set flow-tag,' but not both for a prefix-set.
- Route policy for qos-group and route policy flow-tag cannot have overlapping routes. The Quality-of-service Policy Propagation Using Border Gateway Protocol (QPPB) and flow tag features can coexist (on same as well as on different interfaces) as long as the route policy used by them do not have any overlapping route.
- Mixing usage of qos-group and flow-tag in route-policy and policy-map is not recommended.

Configure Policy-Based Routing

PBR configuration includes the following steps:

1. Configure Flow-tag
2. Provision Forward Class using RPL
3. Configure ACLs with Policy-Based Routing

Configure Flow-tag

The following is a sample definition of a named AS path set in the route policy:

```
as-path-set as-set-1
   ios-regex '_12$',
   ios-regex '_13$'
end-set
```

Use the following sample configurations to set the Flow-tag, and apply the related configuration.

```
/* Set the flow-tag under route-policy configuration */
```

```
Router(config)# route-policy flowtag_match
Router(config-rpl)# if community matches-every (100:1) then
   set flow-tag 101
else
Router(config-rpl-else)# if as-path in as-set-1 then
   set flow-tag 121
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy

/* Apply the policy when updating the routing table */

Router(config)# router bgp 100
Router(config-bgp)# bgp router-id 209.165.201.19
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy flowtag_match

/* Configure a class map */

Router(config)# class-map type traffic match-all green-tag1
Router(config-cmap)# match flow-tag 101
Router(config-cmap)# end-class-map
Router(config-cmap)# exit
Router(config)# policy-map type pbr nh_select
Router(config-pmap)# class type traffic green-tag1
Router(config-pmap-c)# set forward-class 1

/* Configure an interface and apply the PBR policy map to the interface */

Router(config)# interface HundredGigE0/7/0/27
Router(config-if)# ipv4 address 10.10.20.10
Router(config-if)# service-policy type pbr input nh_select
```

Running Configuration

```
Router# show running-config class-map
class-map type traffic match-all green-tag1
match flow-tag 101
 end-class-map
!


Router# show running-config policy-map
policy-map type pbr nh_select
class type traffic green-tag1
  redirect ipv4 nexthop 10.1.2.2
 !
 class type traffic green-tag1
 !
 end-policy-map
!


Router# show running-config interface HundredGigE0/7/0/27
interface HundredGigE0/7/0/27
service-policy type pbr input nh_select
!
```

Provision Forward Class using RPL

Use the following sample configuration to provision Forward Class using RPL.

```
/* Set the forward class ID for community string */

Router(config)# route-policy c1
Router(config-rpl)# if community matches-every (6500:1) then
   set forward-class 1
Router(config-rpl-else)# endif
Router(config-rpl)# end-policy

Router(config)# router bgp 50
Router(config-bgp)# bgp router-id 209.165.201.29
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy c1
Router(config-bgp-af)# exit
Router(config-bgp)# exit

Router(config)# interface tunnel-te1
Router(config-if)# forward-class 1
```

In this example, BGP on the receiving PE is configured with a table policy, and all routes that matches the community string are tagged with a forward-class-id of 1.

```
/* Set the forward class ID for VRF */

Router(config)# route-policy c1
Router(config-rpl)# set forward-class 1
Router(config-rpl)# end-policy

Router(config)# route-policy c2
Router(config-rpl)# set forward-class 2
Router(config-rpl)# end-policy

Router(config)# router bgp 50
Router(config-bgp)# bgp router-id 209.165.201.29
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# exit
Router(config-bgp)# exit

Router(config-bgp)# vrf one
Router(config-bgp-vrf)# rd 1:1
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy c1
Router(config-bgp-af)# exit
Router(config-bgp)# exit

Router(config-bgp)# vrf two
Router(config-bgp-vrf)# rd 2:2
Router(config-bgp-vrf)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy c2
Router(config-bgp-af)# exit
Router(config-bgp)# exit
```

```
Router(config)# interface tunnel-te1
Router(config-if)# forward-class 1

Router(config)# interface tunnel-te2
Router(config-if)# forward-class 2
```

In this example, BGP on receiving PE is configured with a table-policy (C1) and (C2) for two different VRFs.

The policy (C1) sets forward-class 1. From the available TE paths, tunnel-te1 with forward-class 1 is selected for forwarding. Similarly for VRF two, traffic tunnel-te2 associated with forward-class 2 is selected for forwarding.

```
/*Configure a route policy with next hop and set a forward class */

Router(config)# prefix-set nh-set-1
Router(config-pfx)# 10.10.0.1
Router(config-pfx)# end-set

Router(config)# route-policy c1
Router(config-rpl)# if next-hop in nh-set-1 then
   set forward-class 1
Router(config-rpl)# endif
Router(config-rpl)# end-policy

Router(config)# router bgp 50
Router(config-bgp)# bgp router-id 209.165.201.29
Router(config-bgp)# address-family ipv4 unicast
Router(config-bgp-af)# table-policy c1
Router(config-bgp-af)# exit
Router(config-bgp)# exit

Router(config)# interface tunnel-te1
Router(config-if)# forward-class 1
```

In this example, BGP on receiving PE is configured with a table-policy (C1), and the policy (C1) sets the forward-class. From the available TE paths, tunnel-te1 with forward-class 1 is selected for forwarding.

Configure ACLs with Policy-Based Routing

Use the following sample configuration to configure ACLs with PBR.

```
/* Configure an access list */
Router(config)# ipv4 access-list INBOUND-ACL
Router(config-ipv4-acl)# 10 permit ipv4 any host 10.1.1.10
Router(config-ipv4-acl)# 20 permit ipv4 any host 10.2.3.4
Router(config-ipv4-acl)# commit
Mon Nov  6 17:22:42.529 IST
Router(config-ipv4-acl)# exit

/* Configure a class map for the access list */
Router(config)# class-map type traffic match-any INBOUND-CLASS
Router(config-cmap)# match access-group ipv4 INBOUND-ACL
Router(config-cmap)# end-class-map
Router(config)# commit
Mon Nov  6 17:29:12.026 IST
```

```
/* Configure a PBR policy map with the class map */
Router(config)# policy-map type pbr INBOUND-POLICY
Router(config-pmap)# class type traffic INBOUND-CLASS
Router(config-pmap-c)# redirect ipv4 nexthop 192.168.10.1
Router(config-pmap-c)# exit
Router(config-pmap)# class type traffic class-default
Router(config-pmap-c)# transmit
Router(config-pmap-c)# commit
Mon Nov  6 17:25:33.858 IST
Router(config-pmap)# end-policy-map

/* Configure a GigE interface and apply the PBR policy map to the interface */
Router(config)# interface GigabitEthernet 0/0/0/0
Router(config-if)# ipv4 address 10.10.10.1 255.255.255.0
Router(config-if)# service-policy type pbr input INBOUND-POLICY
Router(config-if)# commit
Mon Nov  6 17:31:23.645 IST
Router(config-if)# exit
```

Running Configuration

```
Router# show running-config ip access-list
ipv4 access-list INBOUND-ACL
10 permit ipv4 host 10.10.10.1 any
!
```