Chapter 10

Configuring CoS-Based Forwarding

CoS-based forwarding (CBF) enables you to control next-hop selection based on a packet's class of service and, in particular, the value of the IP packet's precedence bits.

For example, you might want to specify a particular interface or next hop to carry high-priority traffic while all best-effort traffic takes some other path. When a routing protocol discovers equal-cost paths, it can pick a path at random or load-balance across the paths through either hash selection or round robin. CBF allows path selection based on class.

To configure CBF properties, you can include the following statements at the [edit class-of-service] hierarchy level of the configuration:

```
class-of-service {
    forwarding-policy {
        next-hop-map map-name {
            forwarding-class class-name {
                next-hop [ next-hop-name ];
                lsp-next-hop [ lsp-regular-expression ];
            }
        }
        class class-name {
            classification-override {
                forwarding-class class-name;
            }
        }
    }
}
```

This chapter discusses the following topics:

Configuring CoS-Based Forwarding

You can apply CBF only to a defined set of routes. Therefore you must configure a policy statement as in the following example:

```
[edit]
policy-options {
    policy-statement my-cos-forwarding {
        from {
            route-filter filter-name;
        }
        then {
            cos-next-hop-map map-name;
        }
    }
}
```

This configuration specifies that routes matching the route filter will be subject to the CoS next-hop mapping specified by *map-name*. For more information about configuring policy statements, see the *JUNOS Policy Framework Configuration Guide*.

To specify a CoS next-hop map, include the forwarding-policy statement at the [edit class-of-service] hierarchy level:

```
[edit class-of-service]
forwarding-policy {
    next-hop-map map-name {
        forwarding-class class-name {
            next-hop [ next-hop-name ];
            lsp-next-hop [ lsp-regular-expression ];
        }
    }
}
```

When you configure CBF with OSPF as the interior gateway protocol (IGP), you must specify the next hop as an interface name or next-hop alias, not as an IP address. This is true because OSPF adds routes with the interface as the next hop for point-to-point interfaces; the next hop does not contain the IP address. For an example configuration, see “Example: Configuring CoS-Based Forwarding” on page 105.

For Layer 3 VPNs, when you use class-based forwarding for the routes received from the far-end PE router within a VRF instance, the software can match the routes based on the attributes that come with the received route only. In other words, the matching can be based on the route within RIB-in. In this case, the route-filter statement you include at the [edit policy-options policy-statement my-cos-forwarding from] hierarchy level has no effect because the policy checks the bgp.l3vpn.0 table, not the *vrf*.inet.0 table.

The JUNOS software applies the CoS next-hop map to the set of next hops previously defined; the next hops themselves can be located across any outgoing interfaces on the routing platform. For example, the following configuration associates a set of forwarding classes and next-hop identifiers:

```
[edit class-of-service forwarding-policy]
next-hop-map map1 {
        forwarding-class expedited-forwarding {
           next-hop next-hop1;
           next-hop next-hop2;
        }
        forwarding-class best-effort {
           next-hop next-hop3;
           lsp-next-hop lsp-next-hop4;
      }
}
```

In this example, next-hop*N* is either an IP address or an egress interface for some next hop, and lsp-next-hop4 is a regular expression corresponding to any next hop with that label. Q1 through Q*N* are a set of forwarding classes that map to the specific next hop. That is, when a packet is switched with Q1 through Q*N*, it will be forwarded out the interface associated with the associated next hop.

This configuration has the following implications:

- A single forwarding class can map to multiple standard next hops or LSP next hops. This implies that load sharing is done across standard next hops or LSP next hops servicing the same class value. To make this work properly, the JUNOS software creates a list of the equal-cost next hops and forwards packets according to standard load-sharing rules for that forwarding class.

- If a forwarding class configuration includes LSP next hops and standard next hops, the LSP next hops are preferred over the standard next hops. In the preceding example, if both next-hop3 and lsp-next-hop4 are valid next hops for a route to which map1 is applied, the forwarding table includes entry lsp-next-hop4 only.

- If next-hop-map does not specify all possible forwarding classes, the default forwarding class is selected as the default. If the default forwarding class is not specified in the next-hop map, a default is designated randomly. The default forwarding class is the class associated with queue 0.

- For LSP next hops, the JUNOS software uses UNIX regex(3)-style regular expressions. For example, if the following labels exist: lsp, lsp1, lsp2, lsp3, the statement lsp-next-hop lsp matches lsp, lsp1, lsp2, and lsp3. If you do not desire this behavior, you must use the anchor characters lsp-next-hop "^lsp$", which match lsp only.

- The route filter does not work because the policy checks against the bgp.l3vpn.0 table instead of the *vrf*.inet.0 table.

The final step is to apply the route filter to routes exported to the forwarding engine. This is shown in the following example:

```
routing-options {
    forwarding-table {
        export my-cos-forwarding;
    }
}
```

This configuration instructs the routing process to insert routes to the forwarding engine matching my-cos-forwarding with the associated next-hop CBF rules.

The following algorithm is used when you apply a configuration to a route:

- If the route is a single next-hop route, all traffic will go to that route; that is, no CBF will take effect.
- For each next hop, associate the proper forwarding class. If a next hop appears in the route but not in the cos-next-hop map, it will not appear in the forwarding table entry.
- The default forwarding class is used if all forwarding classes are not specified in the next-hop map. If the default is not specified, one is chosen randomly.

Overriding the Input Classification

For IPv4 or IPv6 packets, you can override the incoming classification, assigning them to the same forwarding class based on their input interface, input precedence bits, or destination address. You do so by defining a policy class when configuring CoS properties and referencing this class when configuring a routing policy.

When you override the classification of incoming packets, any mappings you configured for associated precedence bits or incoming interfaces to output transmission queues are ignored. Also, if the packet loss priority (PLP) bit was set in the packet by the incoming interface, the PLP bit is cleared.

To override the input packet classification, do the following:

1. Define the policy class by including the class statement at the [edit class-of-service policy] hierarchy level:

```
[edit class-of-service]
forwarding-policy {
    class class-name {
        classification-override {
            forwarding-class class-name;
        }
    }
}
```

class-name is a name that identifies the class.

2. Associate the policy class with a routing policy by including it in a policy-statement statement at the [edit policy-options] hierarchy level. Specify the destination prefixes in the route-filter statement and the CoS policy class name in the then statement.

```
[edit policy-options]
policy-statement policy-name {
    term term-name {
        from {
            route-filter destination-prefix match-type <class class-name>;
        }
        then class class-name;
    }
}
```

3. Apply the policy by including the export statement at the [edit routing-option] hierarchy level:

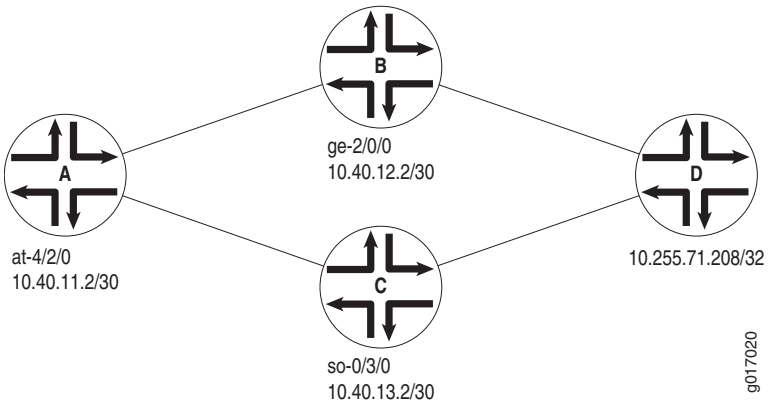
```
[edit routing-options]
forwarding-table {
    export policy-name;
}
```

Example: Configuring CoS-Based Forwarding

Router A has two routes to destination 10.255.71.208 on Router D. One route goes through Router B, and the other goes through Router C, as shown in Figure 3 on page 105.

Configure Router A with CBF to select Router B for queue 0 and queue 2, and Router C for queue 1 and queue 3.

Figure 3: Sample CoS-Based Forwarding



When you configure CBF with OSPF as the IGP, you must specify the next hop as an interface name, not as an IP address. The next hops in this example are specified as ge-2/0/0.0 and so-0/3/0.0.

```
[edit class-of-service]
forwarding-policy {
    next-hop-map my_cbf {
        forwarding-class be {
            next-hop ge-2/0/0.0;
        }
        forwarding-class ef {
            next-hop so-0/3/0.0;
        }
        forwarding-class af {
            next-hop ge-2/0/0.0;
        }
        forwarding-class nc {
            next-hop so-0/3/0.0;
        }
    }
}
classifiers {
    inet-precedence inet {
        forwarding-class be {
            loss-priority low code-points [ 000 100 ];
        }
        forwarding-class ef {
            loss-priority low code-points [ 001 101 ];
        }
        forwarding-class af {
            loss-priority low code-points [ 010 110 ];
        }
        forwarding-class nc {
            loss-priority low code-points [ 011 111 ];
        }
    }
}
forwarding-classes {
    queue 0 be;
    queue 1 ef;
    queue 2 af;
    queue 3 nc;
}
interfaces {
    at-4/2/0 {
        unit 0 {
            classifiers {
                inet-precedence inet;
            }
        }
    }
}
```

```
[edit policy-options]
policy-statement cbf {
    from {
        route-filter 10.255.71.208/32 exact;
    }
    then cos-next-hop-map my_cbf;
}

[edit routing-options]
graceful-restart;
forwarding-table {
    export cbf;
}

[edit interfaces]
traceoptions {
    file trace-intf size 5m world-readable;
    flag all;
}
so-0/3/0 {
    unit 0 {
        family inet {
            address 10.40.13.1/30;
        }
        family iso;
        family mpls;
    }
}
ge-2/0/0 {
    unit 0 {
        family inet {
            address 10.40.12.1/30;
        }
        family iso;
        family mpls;
    }
}
at-4/2/0 {
    atm-options {
        vpi 1 {
            maximum-vcs 1200;
        }
    }
    unit 0 {
        vci 1.100;
        family inet {
            address 10.40.11.2/30;
        }
        family iso;
        family mpls;
    }
}
```

Configuring CoS-Based Forwarding for IPv6

Configure CBF next-hop maps and CBF LSP next-hop maps for IPv6 addresses. The following example shows a CBF next-hop map for IPv6 addresses.

You can configure a next-hop map with both IPv4 and IPv6 addresses, or you can configure separate next-hop maps for IPv4 and IPv6 addresses and include the from family (inet | inet6) statements at the [edit policy-options policy-options policy-statement *policy-name* term *term-name*] hierarchy level to ensure that only next-hop maps of a specified protocol are applied to a specified route.

If you do not configure separate next-hop maps and include the from family (inet | inet6) statements in the configuration, when a route uses two next hops (whether IPv4, IPv6, interface, or LSP next hop) in at least two of the specified forwarding classes, CBF is used for the route; otherwise, the CBF policy is ignored.

1. Define the CBF next-hop map:

```
[edit class-of-service]
forwarding-policy {
    next-hop-map cbf-map {
      forwarding-class best-effort {
        next-hop [ ::192.168.139.38 192.168.139.38 ];
      }
      forwarding-class expedited-forwarding {
        next-hop [ ::192.168.140.5 192.168.140.5 ];
      }
      forwarding-class assured-forwarding {
        next-hop [ ::192.168.145.5 192.168.145.5 ];
      }
      forwarding-class network-control {
        next-hop [ ::192.168.141.2 192.168.141.2 ];
      }
    }
}
```

2. Define the CBF forwarding policy:

```
[edit policy-options]
policy-statement ls {
    then cos-next-hop-map cbf-map;
}
```

3. Export the CBF forwarding policy:

```
[edit routing-options]
forwarding-table {
    export ls;
}
```