Configuring Modular QoS on Link Bundles

This chapter covers the following topics:

QoS on Link Bundles

A bundle is a group of one or more ports that are aggregated together and treated as a single link. The router supports Ethernet interfaces and VLAN interfaces (bundle sub-interfaces) bundles. All QoS features currently supported on physical interfaces, are also supported on all link bundle interfaces. Applying QoS on bundle members is not supported.

Note From Cisco IOS XR Release 7.3.1 onwards, systems with Cisco NC57 line cards running in compatibility mode support QoS over Layer 2 services for:

- Local switching [xconnect or bridging]
- L2 VPN – VPWS

Starting with Cisco IOS XR Release 7.4.1 systems with Cisco NC57 line cards running in native mode support QoS over Layer 2 services for:

- Local switching [xconnect or bridging]
- L2 VPN – VPWS

Restrictions for Link Bundles

- Only Ethernet link bundling is supported.
- A bundle interface can only contain physical interface.
- All links within a single bundle must be configured either to run 802.3ad (LACP) or Etherchannel (non-LACP). Mixed links within a single bundle are not supported.
- MAC accounting is not supported on Ethernet link bundles.

- Maximum number of links supported in each link bundle is 64.
- The maximum number of link bundles supported is 128.

Load Balancing

Load balancing function is a forwarding mechanism to distribute traffic over multiple links based on Layer 3 routing information in the router. Per-destination load balancing isonly supported on the router, where the router is allowed to distribute packets over one of the links in the bundle. When the per-destination load balancing is enabled, all packets for a certain source-destination pair goes through the same link, though there are multiple links available. In other words, per-destination load balancing can ensure that packets for a certain source-destination pair could arrive in order.

Layer 3 Load Balancing on Link Bundles

Layer 3 load balancing for link bundles is done on Ethernet Flow Points (EFPs) and is based on the IPv4 source and destination addresses in the packet. When Layer 3 service-specific load balancing is configured, all egress bundles are load balanced based on the IPv4 source and destination addresses. When packets do not have IPv4 addresses, default load-balancing (based on the MAC SA/DA fields in the packet header) is used.

Configure QoS on Link Bundles

QoS is configured on link bundles in the same way that it is configured on individual interfaces.

Guidelines

- When a QoS policy is applied on a bundle in the egress direction, it's also applied at each member interface.
- When a QoS policy is applied on a bundle (ingress direction), it's replicated at each NPU core.
- If a QoS policy is not applied to a bundle interface, both the ingress and egress traffic use the default queue of the per link member port.
- The shape rate that is specified in the bundle policy-map is not an aggregate for all bundle members. The shape rate applied to the bundle depends on the load balancing of the links. For example, if a policy map with a shape rate of 10 Mbps is applied to a bundle with two member links, and if the traffic is always load-balanced to the same member link, then an overall rate of 10 Mbps applies to the bundle. However, if the traffic is load-balanced evenly between the two links, the overall shape rate for the bundle becomes 20 Mbps.
- If a member is deleted from a bundle, the total bundle statistics changes because the statistics that belongs to the detached link is lost.
- The QoS policy that is applied on bundle is inherited to all its member links and the reference bandwidth that is used to calculate shaper/bandwidth is applied as per the physical member interface bandwidth, and not the bundle as a whole.

Configuration Example

You have to accomplish the following to complete the QoS configuration on link bundles:

Note The policy works only if it is applied on the ingress direction. The egress is supported on COS, DEI and MPLS exp marking. So the below policy may not work when it is applied on egress.

1. Creating a class-map
2. Creating a policy-map and specifying the respective class-map
3. Specifying the action type for the traffic

   Refer Attach a Traffic Policy to an Interface for details on step 1, 2 and 3.
4. Creating a link bundle
5. Applying traffic policy to the link bundle

```
/* Configure Ether-Bundle and apply traffic policy */
Router(config)# interface Bundle-Ether 12000
Router(config-if)# mtu 9100
Router(config-if)# service-policy input ingress
Router(config-if)# service-policy output egress
Router(config-if)# ipv4 address 100.12.0.0 255.255.255.254
Router(config-if)# bundle maximum-active links 64
Router(config-if)# commit
```

Running Configuration

This example shows how a traffic policy is applied on an Ethernet link bundle. The policy is applied to all interfaces that are members of the Ethernet link bundle.

```
/* Policy-map */

policy-map ingress
 class inet4-classifier-af1
  set qos-group 1
 !
 class inet4-classifier-af2
  set qos-group 2
 !
 class inet4-classifier-af3
  set qos-group 3
 !
 class inet4-classifier-af4
  set qos-group 4
 !
 class inet4-classifier-be1
  set qos-group 5
 !
 class inet4-classifier-nc1
  set qos-group 6
 !
 class class-default
 !
 end-policy-map
!

/* Ether Bundle */
```

```
interface Bundle-Ether12000
 mtu 9100
 service-policy input ingress
 service-policy output egress
 ipv4 address 100.12.0.0 255.255.255.254
 load-interval 30
 flow ipv4 monitor FMM-V4 sampler SM ingress
 flow ipv6 monitor FMM-V6 sampler SM ingress
 flow mpls monitor FMM-MPLS sampler SM ingress
 ipv4 access-group IPV4ACL_101 ingress
 ipv6 access-group IPV6ACL_101 ingress
!
```

Verification

- Verify that the bundle status is UP.

```
router# show bundle bundle-ether 1200
Wed Dec 16 19:55:49.974 PST

Bundle-Ether12000
  Status:                                  Up
  Local links <active/standby/configured>:   35 / 0 / 35
  Local bandwidth <effective/available>:     3500000000 (3500000000) kbps
  MAC address (source):                    ea3b.745f.c4b0 (Chassis pool)
  Inter-chassis link:                      No
  Minimum active links / bandwidth:        1 / 1 kbps
  Maximum active links:                    64
  Wait while timer:                        2000 ms
  Load balancing:                          Default
  LACP:                                    Operational
    Flap suppression timer:                Off
    Cisco extensions:                      Disabled
    Non-revertive:                         Disabled
  mLACP:                                   Not configured
  IPv4 BFD:                                Not configured

  Port                Device           State        Port ID         B/W, kbps
  ------------------  ---------------  -----------  --------------  ----------
  Hu0/4/0/0           Local            Active       0x8000, 0x0009   100000000
      Link is Active
  Hu0/4/0/1           Local            Active       0x8000, 0x000a   100000000
      Link is Active
- - -
- - -
  Hu0/4/0/35          Local            Active       0x8000, 0x002b   100000000
      Link is Active
```

- Verify the bundle statistics:

```
router# show policy-map interface bundle-ether 12000

Bundle-Ether12000 input: ingress

Class inet4-classifier-af1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          4647401962/21236124455654       26403040
    Transmitted         :          4647401962/21236124455654       26403040
    Total Dropped       :                   0/0                    0
Class inet4-classifier-af2
```

```
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          4502980177/20576584333939       25571493
    Transmitted         :          4502980177/20576584333939       25571493
    Total Dropped       :                   0/0                    0
Class inet4-classifier-af3
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          4647404125/21236213667880       26389086
    Transmitted         :          4647404125/21236213667880       26389086
    Total Dropped       :                   0/0                    0
Class inet4-classifier-af4
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          9291188840/42456120548683       52771168
    Transmitted         :          9291188840/42456120548683       52771168
    Total Dropped       :                   0/0                    0
Class inet4-classifier-be1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          4647413429/21235847852686       26393414
    Transmitted         :          4647413429/21235847852686       26393414
    Total Dropped       :                   0/0                    0
Class inet4-classifier-nc1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :          9294887621/42473100149807       52778258
    Transmitted         :          9294887621/42473100149807       52778258
    Total Dropped       :                   0/0                    0

Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :                   0/0                    0
    Transmitted         :                   0/0                    0
    Total Dropped       :                   0/0                    0

Bundle-Ether12000 output: egress

Class c1
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         16665494532/75878118942463       8760591
    Transmitted         :         16655834643/75834136022017       8760591
    Total Dropped       :             9659889/43982920446          0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 9659889/43982920446
Class c2
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         16665421959/75877849543188       8718687
    Transmitted         :         16665421959/75877849543188       8718687
    Total Dropped       :                   0/0                    0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 0/0
Class c3
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         16665247833/75877509455458       8703470
    Transmitted         :         16665187414/75877234624197       8703470
    Total Dropped       :               60419/274831261            0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 60419/274831261
Class c4
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         33330896131/151755393012945      17470745
    Transmitted         :         33330745421/151754709368565      17470745
    Total Dropped       :              150710/683644380            0
  Queueing statistics
    Queue ID                             : None (Bundle)
```

```
    Taildropped(packets/bytes)           : 150710/683644380
Class c5
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         16878910340/76849791869834       8833394
    Transmitted         :         16878849464/76849514633309       8833394
    Total Dropped       :               60876/277236525            0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 60876/277236525
Class c6
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :         33330898844/151756094112925      17456785
    Transmitted         :         33330752668/151755427708382      17456785
    Total Dropped       :              146176/666404543            0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 146176/666404543
Class c7
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :              244106/79922040             74
    Transmitted         :              244106/79922040             74
    Total Dropped       :                   0/0                    0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 0/0
Class class-default
  Classification statistics          (packets/bytes)     (rate - kbps)
    Matched             :        267075066180/1215993441123215     139917482
    Transmitted         :        267075066180/1215993441123215     139917482
    Total Dropped       :                   0/0                    0
  Queueing statistics
    Queue ID                             : None (Bundle)
    Taildropped(packets/bytes)           : 0/0
```

Related Topics

- QoS on Link Bundles, on page 1

Associated Commands

- bundle maximu-active links
- interface Bundle-Ether

LAG-level Scheduling in Egress Queuing

Table 1: Feature History Table

Feature Name	Release Information	Feature Description
LAG-level Scheduling in Egress Queuing	Release 24.3.1	Introduced in this release on: NCS 5700 fixed port routers; NCS 5500 modular routers (NCS 5700 line cards [Mode: Native]) This release enhances traffic management by introducing Link Aggregation Group (LAG) level scheduling improving shaping granularity and scheduler resource efficiency in egress queues for bundle interfaces. Unlike the previous per-member scheduling, where policies are replicated and applied to each individual link in a bundle, LAG-level scheduling applies egress policies to the entire bundle as a single link enabling more precise traffic shaping. The feature introduces these changes: **CLI:** • **hw-module profile qos lag-scheduler** **YANG Data Models:** • New Xpaths for `Cisco-IOS-XR-um-hw-module-profile-cfg.yang` (see GitHub, YANG Data Models Navigator)

LAG-level Scheduler for Egress Queues

LAG-level scheduling is a traffic management feature that applies egress queue policies to an entire Link Aggregation Group (LAG) as a single link, enhancing shaping granularity. This feature supports

- traffic management policies to be applied at the LAG level rather than per member link, and
- flat QoS, Hierarchical QoS (HQoS), and Egress Traffic Management (ETM) mode queuing.

In traditional traffic management systems, policies are applied to each individual link within a link aggregation group or bundle. This per-member approach not only consumes more scheduling resources but also limits the granularity of traffic shaping. For example, in a 2x100G bundle, the policy applied to the bundle gets replicated to each 100G member link, making it difficult to achieve precise traffic shaping. With LAG-level scheduling, the policy is applied to the entire 200G bundle as a single link, significantly enhancing the shaping granularity, allowing for more precise control over traffic flow and better utilization of scheduler resources.

This table helps you understand the key differences between the LAG-level and per-member scheduling features.

Table 2: LAG-level and Per-member Scheduling Features - Key Differences

Attributes	LAG-level Scheduling	Per-member Scheduling
Policy Application	Applies egress queue policies to the entire bundle as a single link.	Applied policy is replicated for each member link.
Scheduling Hierarchy	Tied to the individual NPU core.	Tied to member port.
Shaping Rate	Supports absolute QoS policy shaping rates only.	Supports both absolute and percentage-based QoS policy shaping rates.
Default Status	Configuration required. See Configure LAG-level Scheduling.	Enabled by default.

Use Case Examples:

- **LAG-level scheduling in a single NPU core**—In the traditional bundle-level scheduling, a 2x100G bundle, the policy is replicated to each 100G member link. For a 10-member bundle, traffic is distributed among the members, and a 5 Gbps shaper would effectively become 50 Gbps (5 Gbps x 10).

  With this feature, the service policy is applied to the entire bundle as a single interface. For a 10-member bundle, a 5 Gbps shaper limits the entire bundle to 5 Gbps, providing more granular control. This allows network operators to apply precise rate limits for each customer, ensuring better traffic management

- **LAG-level scheduling on members from different NPU cores**—Consider a scenario with two bundle links on different NPU cores and an 8 Gbps shaper. The resultant shaping would be 16 Gbps (8 Gbps x 2). If a third interface from another NPU core is added, the shaping would be 24 Gbps (8 Gbps x 3). This demonstrates the scalability and efficiency of LAG-level scheduling across multiple NPU cores, providing more granular shaping.

LAG-level Scheduling: Usage Guidelines and Restrictions

Usage Guidelines:

After configuring the LAG-level scheduling feature, you must reload the router to activate the profile.

Restrictions:

- LAG-level scheduling is supported for members from different NPU cores with each core having a separate LAG-level TM scheduler. When a LAG contains link members from multiple NPU cores, each core independently manages its portion of the LAG using its own scheduler.

- Only absolute QoS policy shaping rates are supported; percentage rate configurations are not supported in the LAG scheduling mode.
- Default mode (per member scheduling) and LAG-level scheduler mode cannot be used together.
- Port-level burst is not supported for bundle-ethernet (BE) interfaces.
- Port-level shapers are not supported.
- Low-rate shaper enhancement of 0 kbps for shared shapers is not supported with LAG-level scheduling - minimum shaping granularity is 3.9 Mbps.
- Member pinning is not supported, as it aims to avoid limiting a subscriber to a specific member link.

Configure LAG-level Scheduling

Before you begin

Verify the NCS 5700 line card is operating in the native mode.

Procedure

Step 1 Configure LAG-level scheduling for egress queuing.

Example:

```
Router(config)#hw-module profile qos lag-scheduler
router(config)#commit
```

Step 2 Reload the router for the configuration to take effect.

Example:

```
Router#reload location all
```

Step 3 Verify LAG-level scheduling is configured successfully on a bundle interface.

Example:

```
Router#show ofa objects global dump-without-base location 0/1/CPU0 | i lag_scheduler_enable
Mon Aug 26 11:42:09.531 UTC
uint32_t lag_scheduler_enable => 1
```

In the show command output, the `lag_scheduler_enable` parameter is set to `1`, which specifies that the LAG-level scheduling mode is active.