Understanding Queue Limits and Output Drops on Cisco IOS Software Platforms

Contents

Introduction

This document is applicable to Cisco IOS® Software platforms only, which generally includes Cisco 7200VXR and Cisco ISR 3800, 2800, 1800 series routers, as well as legacy Cisco Access Routers including 3700, 3600, 2600, and 1700 series routers.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Cisco IOS Quality of Service Solutions Configuration Guide
- QoS---Hierarchical Queueing Framework (HQF)

Components Used

The information in this document is based on these software versions:

- For Pre-HQF: Cisco routers that run Cisco IOS Software Release 12.3(26)
- For HQF: Cisco routers that run Cisco IOS Software Release 12.4(22)T

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Conventions

Refer to Cisco Technical Tips Conventions for more information on document conventions.

Class-Based Weighted Fair Queueing Primer

In pre-HQF IOS images, generally speaking, any class with a **bandwidth** command will generally be prioritized against classes without *bandwidth* or *priority* based on the classes' *Weight*. In order to understand the Class-Based Weighted Fair Queueing (CBWFQ) scheduling algorithm, you must first understand the concept of a weight, which is flow-specific for Flow Based Fair Queues and Class specific for individual Class Based Queues within the Class Based Weighted Fair Queue.

The formula to derive the weight for a Flow Based Fair Queue is:

```
32384 / (IP Prec + 1)
```

User defined classes within a class-based Weighted Fair queue derive their respective weights as a function of the **bandwidth** command configured in the class as a proportion of the SUM of all *bandwidth* classes in the Class Based Queue. The exact formula is proprietary.

In HQF images, flow-based fair-queues, configurable in both user-defined classes and class default with *fair-queue*, are scheduled equally (instead of by Weight). Furthermore, in HQF, a Class Based Queue's scheduling priority is determined based on the HQF scheduler and not on the classes' legacy Weight formula.

Note: This section is not intended to be a comprehensive behavioral analysis for Class Based Queueing operations. The intention is a brief education as CBWFQ applies to understanding queue limits and Output Drops.

Understanding Queue Limits and Output Drops

User-Defined Classes Configured with the "priority" Command

For MQC User-Defined classes configured with any variant of the **priority** command, including **priority**, **priority <*kbps*>** , and **priority percent**.

Pre-HQF Behavior

Technically, even though no CLI exists to see it, and it is not configurable, a "hidden" system queue exists which is shared by all priority class data. This acts as a holding area for priority data after it has been classified and after it has been permitted by the congestion-aware policer. LLQ packets are placed in this "hidden" system queue if they cannot be placed directly on the egress interface's transmit ring during the receive interrupt, which is otherwise called functional congestion. In this situation, because functional congestion exists, the packet is evaluated against the LLQ conditional policer during the receive interrupt while still owned by the receiving interface's driver. If the packet is not dropped by the LLQ's conditional policer, it is placed in this "hidden" LLQ system queue and the receive interrupt is released. Therefore, all packets placed in this "hidden" system queue have conformed to the LLQ's congestion aware policer and will be dequeued to the egress interface's transmit ring immediately by the LLQ/CBWFQ scheduler.

Despite this queue's existence, the behavior is unlike IOS queues created for non-LLQ data (such

as fair-queue and bandwidth queues) in that no additional queueing latency (above the transmit-ring latency) will be incurred since packets in this queue will be immediately drained to the transmit ring by LLQ/CBWFQ scheduler. If a priority class packet is not dropped by the conditional policer during the receive interrupt, then this LLQ packet will exist in this “hidden” system queue briefly before dequeueing to the egress interface’s transmit ring. In this case, the LLQ/CBWFQ scheduler will immediately present the packet to the egress interface’s transmit ring. The Conditional Policer has run before admitting the packet to the LLQ/CBWFQ, thereby limiting the LLQ to the configured priority rate.

In summary, it is recommended to drop LLQ data that exceeds the priority rate during times of congestion than to incur additional queueing latency, which is a fundamental component of LLQ. This Conditional policing mechanism permits a strict priority queue without allowing the priority queue to monopolize the entire interface PLIM, which is an improvement over IOS’s legacy Priority Queueing Feature.

- **Pre-HQF queue limit**: NA
- **Pre-HQF “priority” + “random-detect” behavior**: NA, WRED not allowed in LLQ.
- **Pre-HQF “priority” + “fair-queue” behavior**: NA, fair-queue not allowed in LLQ.
- **Pre-HQF “priority” + “random-detect” + “fair-queue” behavior**: NA, neither fair-queue or random-detect supported in LLQ.

HQF Behavior

Same as Pre-HQF except the hidden queue is no longer hidden and the queue-limit is now configurable and defaults to 64 packets.

- **HQF queue limit**: 64 packets
- **HQF “priority” + “random-detect” behavior**: NA, WRED not allowed in LLQ.
- **HQF “priority” + “fair-queue” behavior**: NA, fair-queue not allowed in LLQ.
- **HQF “priority” + “random-detect” + “fair-queue” behavior**: NA, neither fair-queue or random-detect supported in LLQ.

User Defined Classes Configured with the "bandwidth" Command

For MQC User Defined classes configured with any variant of the **bandwidth** command, including **bandwidth <*kbps*>** , **bandwidth percent**, and **bandwidth remaining percent**.

Pre-HQF Behavior

The default queue-limit is 64 packets, which is tunable. If, during the receive interrupt, you need to enqueue a packet which would result in > 64 packets in the queue, the packet is tail dropped.

- **Pre-HQF queue-limit**: 64 packets, tunable via queue-limit.
- **Pre-HQF “bandwidth” + “random-detect” behavior**:Example:

```
policy-map PRE_HQF_BANDWIDTH_WRED
 class 1
   bandwidth 32
   random-detect
```

  If any variant of bandwidth is configured along with any variant of random-detect, ignore any queue-limit CLI, which effectively removes any buffer limit in the class. In other words,

random-detect and queue-limit are mutually exclusive in pre-HQF images. Using random-detect as drop strategy, the current queue size is unrestrained and can theoretically occupy every buffer allocated to the class based fair-queue, where this number of buffers allocated to the class based fair-queue is derived based on service-policy attach point:Physical interface: 1000 packets, tunable with interface CLI hold-queue outATM PVC: 500 packets, tunable with PVC CLI vc-hold-queueFrame-Relay map-class: 600 packets, tunable with frame-relay map-class CLI frame-relay holdqClass-Based shaping policy applied to (sub)interface (pre-HQF): 1000 packets, tunable with MQC class CLI shape max-buffers.**Note:** All Frame-Relay and Class Based shaping examples assume the sum of shapers does not exceed interface clock rate.**Note:** In pre-HQF images, when a Class Based Shaping policy is applied to a (sub)interface, beware of the underlying physical interface's speed, as interfaces <2Mbps will default to a Weighted Fair Queue and interfaces >2Mbps will default to FIFO. In pre-HQF, the shaping queue will feed the interface hold queue, whether the shaping policy is attached at the subinterface or the physical interface level.During the receive interrupt, each time a packet becomes a candidate for an interface's output queue, the WRED average queue size is calculated using this formula:

```
Average Queue Size = (old_average * (1-1/2^n)) + (current_queue_size * 1/2^n)
```

If the resultant Average Queue Size is:Smaller than the WRED min-threshold, enqueue the packet and release the receive interrupt.Between the WRED min-threshold and WRED max-threshold, possibly drop the packet with increased probability as the Average Queue Size gets closer to WRED max-threshold. If the Average Queue Size is exactly equal to WRED max-threshold, the packet is dropped according to the mark probability denominator. The mark probability denominator also serves as a baseline to determine what percentage of packets will get dropped when the Average Queue Limit is not exactly equal to WRED max-threshold but is higher than WRED min-threshold. This is a graphical example:If the packet is dropped, the receive interrupt is released and a Random drop is incremented. If the packet is not dropped, the packet is enqueued and the receive interrupt is released.Higher than the WRED max-threshold, drop the packet, release the receive interrupt, and increment a Tail Drop.

- **Note:** IP Precedence based (default) and DSCP-based WRED allow the min-threshold, max-threshold, and mark probability denominator to be defined differently for different values. This is where the Weighted component of Random Early Detection is evident. You can protect certain ToS values relative to other values via tuning their relative thresholds and mark probability denominators.When random detect and bandwidth are configured together, the Current Queue Size can be larger than the WRED max-threshold at any given point in time. This is because WRED minimum and maximum thresholds only take action based on the Average (not Current) Queue Size. This provides an opportunity to expire all buffers allocated to the Class Based Queue which could result in no-buffer drops occurring anywhere within the Class Based Fair Queue (refer to Cisco bug ID CSCsm94757).
- **Pre-HQF "bandwidth" + "fair-queue" behavior:** NA, fair-queue not allowed in bandwidth class.
- **Pre-HQF "bandwidth" + "random-detect" + "fair-queue" behavior**: NA, fair-queue not allowed in bandwidth class

HQF Behavior

The behavior is the same as described in the Pre-HQF section.

- **HQF queue-limit**: 64 packets, tunable via queue-limit.This is same as that in the pre-HQF.

- **HQF “bandwidth” + “random-detect” behavior**:Example:

```
policy-map HQF_BANDWIDTH_WRED
 class 1
   bandwidth 32
   queue-limit 512
   random-detect
```

Note: The default queue-limit is 64 packets.The behavior is the same as in corresponding pre-HQF section, with one important exception. In HQF images, random-detect and queue-limit can co-exist in the same User-Defined class (or class class-default) and queue-limit will be enabled and tuned to 64 packets in a default configuration. As such, queue-limit will serve as a maximum current queue size in a random-detect class, therefore providing a mechanism to limit no-buffer drops discussed in the corresponding pre-HQF section. Due to this addition, the configured queue-limit must be at least as large as the random-detect max-threshold, where the random-detect max-threshold will default to 40 packets, or else the parser will reject the configuration.This introduces a current-queue-limit check in WRED classes, whereby, even if Average Queue Depth calculation is smaller than max-threshold, if the Current (not Average) Queue Size is greater than the queue-limit, the packet will be dropped, the receive interrupt released, and a Tail Drop recorded. Remember, if the queue-limit is tuned sufficiently high to exhaust the aggregate queueing buffers for the Class-Based Queue, no-buffer drops can still occur. Aggregate queueing buffers for HQF are defined here:Physical interface: 1000 packets, tunable with interface CLI hold-queue outATM PVC: 500 packets, tunable with PVC CLI vc-hold-queueFrame-Relay map-class: 600 packets, tunable with frame-relay map-class CLI frame-relay holdqClass-Based shaping policy applied to physical interface in HQF code: 1000 packets, tunable with a combination of interface CLI hold-queue out and child policy queue-limit where child policy queue-limit has an upper bound of interface hold-queue out.Class-Based shaping policy applied to subinterface in HQF code: 512 packets, not tunable (investigate with NSSTG QoS platform Team, should it be tunable)**Note:** All Frame-Relay and Class Based shaping examples assume the sum of shapers does not exceed interface clock rate.Here is a real world example:

```
policy-map JACKLYN
 class 1
    bandwidth 64
    queue-limit 500 packets
     random-detect
     random-detect precedence 1 22 300
```

During this output, no traffic is being generated through the interface:

```
F340.11.25-7200-5_LAC#show policy-map interface | i queue
      queue limit 500 packets
      (queue depth/total drops/no-buffer drops) 0/387595/0
!--- Current_q_depth is 0 Mean queue depth: 107 packets !--- last calculation of
Average_queue_depth
```

At this point, traffic is started. The stream is non-bursty at 400PPS, consisting of1000 byte frames:

```
F340.11.25-7200-5_LAC#show policy-map interface | i queue
      queue limit 500 packets
      (queue depth/total drops/no-buffer drops) 461/387641/0
!--- 461 is Current_q_depth > Prec 1 max-thresh of 300 !--- but < "queue-limit 500 packets".
Mean queue depth: 274 packets !--- Avg_q_depth is rising, Mark Prob Denom is being used.
F340.11.25-7200-5_LAC#show policy-map interface | i queue queue limit 500 packets (queue
depth/total drops/no-buffer drops) 363/387919/0 !--- 363 is Current_q_depth and it is
falling compared to last !--- iteration because WRED is random dropping packets. Mean queue
depth: 351 packets !--- Avg_q_depth is now above max_thresh, WRED starts to tail-drop !---
in addition to random-drop. F340.11.25-7200-5_LAC#show policy-map interface | i queue queue
limit 500 packets (queue depth/total drops/no-buffer drops) 199/388263/0 Mean queue depth:
312 packets F340.11.25-7200-5_LAC#show policy-map interface | i queue queue limit 500
```

```
packets (queue depth/total drops/no-buffer drops) 303/388339/0 Mean queue depth: 276 packets
F340.11.25-7200-5_LAC#show policy-map interface | i queue queue limit 500 packets (queue
depth/total drops/no-buffer drops) 325/388498/0 Mean queue depth: 314 packets F340.11.25-
7200-5_LAC#show policy-map interface | i queue queue limit 500 packets (queue depth/total
drops/no-buffer drops) 298/390075/0 Mean queue depth: 300 packets
```

Notice how, eventually, with a non-bursty stream, the WRED Average Queue Depth will equal the Current Queue Depth, which is the expected behavior.

- **HQF “bandwidth” + “fair-queue” behavior**:When bandwidth and fair-queue are applied together to an HQF User Defined class, each flow-based queue is allocated a queue-limit equal to .25 * queue-limit. Because the default queue-limit is 64 packets, each flow based queue in a fair-queue will be allocated 16 packets. If four flows were traversing this class, by default each flow-queue would have 16 packets, therefore you would never expect to see total packets enqueued of >64 (4 *16). All tail drops from an individual flow-queue are recorded as flowdrops. If the number of flow-queues were significantly high as was the queue-limit, then another opportunity for no-buffer drops. For example, assuming policy attach-point is a physical interface, where 1000 aggregate buffers are allocated:

```
policy-map TEST
 class 1
  bandwidth 32
  fair-queue 1024
  queue-limit 128
```

  In this configuration, appreciable traffic in all flow queues can starve aggregate interface buffers and result in no-buffer drops in other User-Defined classes (see Cisco bug ID CSCsw98427). This is because 1024 flow queues, each with a 32 packet queue-limit can easily oversubscribe the 1000 aggregate interface Class Based Queuing buffer allocation.
- **HQF “bandwidth” + “random-detect” + “fair-queue” behavior**:Example:

```
policy-map TEST
 class 1
  bandwidth 32
  fair-queue 1024
  queue-limit 128
  random-detect
```

  Same as bandwidth and fair-queue in section except WRED Average Queue Size is calculated every time a packet arrives to decide whether the packet should be random dropped or tail dropped. As with pre-HQF, all flow-queues will share one instance of WRED thresholds, meaning all packets enqueued to all flow-queues are used to calculate WRED Average Queue Depth, then the drop decision applies the WRED minimum and maximum thresholds against the aggregate packets in all flow queues. However, another departure from bandwidth and fair-queue in section, because one instance of WRED thresholds apply to all flow-based queues, the individual flow-queues’ queue-limit (.25 * “queue-limit”) is ignored and instead honor the Classes aggregate queue-limit for a Current Queue Limit check.

Class Default Behavior

Pre-HQF Behavior

In pre-HQF, Class Default defaults to fair-queue, all flow-queues share the queue-limit for the class (default is 64), and there is no bandwidth reservation. In other words, the total number of packets enqueued in all flow-queues will never exceed the queue-limit. The amount of packets enqueued in each flow-queue will depend on the calculated Weight of the flow-queue. Conversely, if fair-queue and random-detect are used together in class-default, the queue-limit will be ignored and all flow-queues will share the same WRED thresholds. As such, all packets currently

enqueued in all flow-queues will be used to calculate the WRED Average Queue Size. Because the Current Queue Size has no upper limit in this configuration, the opportunity for no-buffer drops is high (refer to Cisco bug ID CSCsm94757).

- Adding bandwidth to class-default will result in behavior outlined in the Pre-HQF Behavior - User Defined Classes Configured with the "bandwidth" Command section.
- Adding bandwidth and random-detect to class class-default will result in behavior outlined in the *Pre-HQF "bandwidth" + "random-detect" behavior* section of Pre-HQF Behavior - User Defined Classes Configured with the "bandwidth" Command.

Note: In pre-HQF, bandwidth cannot co-exist with fair-queue in class-default.

HQF Behavior

In HQF, Class Default defaults to a FIFO queue and is allocated a pseudo bandwidth reservation based on the leftover allocations from User Defined Classes. As such, for HQF class class-default default behavior, see the HQF Behavior - User Defined Classes Configured with the "bandwidth" Command section. At all times, regardless of configuration, class class-default in HQF images will always have an implicit bandwidth reservation equal to the unused interface bandwidth not consumed by user-defined classes. By default, the class-default class receives a minimum of 1% of the interface or parent shape bandwidth. It is also possible to explicitly configure the bandwidth CLI in class default.

- If fair-queue is configured in class Class-Default, the behavior matches the *HQF "bandwidth" + "fair-queue" behavior* section of HQF Behavior - User Defined Classes Configured with the "bandwidth" Command.
- If fair-queue and random-detect are configured together in Class-Default, the behavior matches the *HQF "bandwidth" + "random-detect" + "fair-queue" behavior* section of HQF Behavior - User Defined Classes Configured with the "bandwidth" Command.

Related Information

- **Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2SR**
- **QoS---Hierarchical Queueing Framework (HQF)**
- **QoS Technology Support**
- **Routers Product Support**
- **Technical Support & Documentation - Cisco Systems**