QoS: Congestion Avoidance Configuration Guide, Cisco IOS XE Release 2

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

Congestion Avoidance Overview

Congestion avoidance techniques monitor network traffic loads in an effort to anticipate and avoid congestion at common network bottlenecks. Congestion avoidance is achieved through packet dropping. Among the more commonly used congestion avoidance mechanisms is Random Early Detection (RED), which is optimum for high-speed transit networks. Cisco IOS XE Software includes an implementation of RED, called Weighted RED (WRED), that combines the capabilities of the RED algorithm with the IP Precedence feature. WRED, when configured, controls when the router drops packets.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Weighted Random Early Detection

WRED helps avoid the globalization problems that can occur. Global synchronization occurs as waves of congestion crest only to be followed by troughs during which the transmission link is not fully utilized. Global synchronization of TCP hosts, for example, can occur because packets are dropped all at once. Global synchronization manifests when multiple TCP hosts reduce their transmission rates in response to packet dropping and then increase their transmission rates once again when the congestion is reduced.

About Random Early Detection

The RED mechanism was proposed by Sally Floyd and Van Jacobson in the early 1990s to address network congestion in a responsive rather than reactive manner. Underlying the RED mechanism is the premise that most traffic runs on data transport implementations that are sensitive to loss and will temporarily slow down when some of their traffic is dropped. TCP, which responds appropriately--even robustly--to traffic

drop by slowing down its traffic transmission, effectively allows the traffic-drop behavior of RED to work as a congestion-avoidance signalling mechanism.

TCP constitutes the most heavily used network transport. Given the ubiquitous presence of TCP, RED offers a widespread, effective congestion-avoidance mechanism.

In considering the usefulness of RED when robust transports such as TCP are pervasive, it is important to consider also the seriously negative implications of employing RED when a significant percentage of the traffic is not robust in response to packet loss. Neither Novell NetWare nor AppleTalk is appropriately robust in response to packet loss, therefore you should not use RED for them.

How It Works

RED aims to control the average queue size by indicating to the end hosts when they should temporarily slow down transmission of packets.

RED takes advantage of the congestion control mechanism of TCP. By randomly dropping packets prior to periods of high congestion, RED tells the packet source to decrease its transmission rate. Assuming the packet source is using TCP, it will decrease its transmission rate until all the packets reach their destination, indicating that the congestion is cleared. You can use RED as a way to cause TCP to slow down transmission of packets. TCP not only pauses, but it also restarts quickly and adapts its transmission rate to the rate that the network can support.

RED distributes losses in time and maintains normally low queue depth while absorbing spikes. When enabled on an interface, RED begins dropping packets when congestion occurs at a rate you select during configuration.

For an explanation of how the Cisco WRED implementation determines parameters to use in the WRED queue size calculations and how to determine optimum values to use for the weight factor, see the section Average Queue Size, page 6 later in this module.

Packet Drop Probability

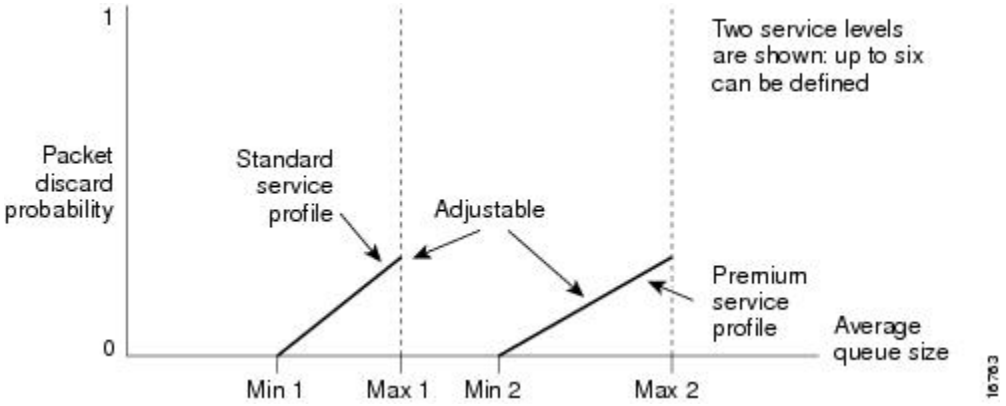
The packet drop probability is based on the minimum threshold, maximum threshold, and mark probability denominator.

When the average queue depth is above the minimum threshold, RED starts dropping packets. The rate of packet drop increases linearly as the average queue size increases until the average queue size reaches the maximum threshold.

The mark probability denominator is the fraction of packets dropped when the average queue depth is at the maximum threshold. For example, if the denominator is 512, one out of every 512 packets is dropped when the average queue is at the maximum threshold.

When the average queue size is above the maximum threshold, all packets are dropped. The figure below summarizes the packet drop probability.

Figure 1 RED Packet Drop Probability



The minimum threshold value should be set high enough to maximize the link utilization. If the minimum threshold is too low, packets may be dropped unnecessarily, and the transmission link will not be fully used.

The difference between the maximum threshold and the minimum threshold should be large enough to avoid global synchronization of TCP hosts (global synchronization of TCP hosts can occur as multiple TCP hosts reduce their transmission rates). If the difference between the maximum and minimum thresholds is too small, many packets may be dropped at once, resulting in global synchronization.

How TCP Handles Traffic Loss

Note The sections How TCP Handles Traffic Loss, page 3 and How the Router Interacts with TCP, page 4 contain detailed information that you need not read in order to use WRED or to have a general sense of the capabilities of RED. If you want to understand why problems of global synchronization occur in response to congestion and how RED addresses them, read these sections.

When the recipient of TCP traffic--called the receiver--receives a data segment, it checks the four octet (32-bit) sequence number of that segment against the number the receiver expected, which would indicate that the data segment was received in order. If the numbers match, the receiver delivers all of the data that it holds to the target application, then it updates the sequence number to reflect the next number in order, and finally it either immediately sends an acknowledgment (ACK) packet to the sender or it schedules an ACK to be sent to the sender after a short delay. The ACK notifies the sender that the receiver received all data segments up to but not including the one marked with the new sequence number.

Receivers usually try to send an ACK in response to alternating data segments they receive; they send the ACK because for many applications, if the receiver waits out a small delay, it can efficiently include its reply acknowledgment on a normal response to the sender. However, when the receiver receives a data segment out of order, it immediately responds with an ACK to direct the sender to resend the lost data segment.

When the sender receives an ACK, it makes this determination: It determines if any data is outstanding. If no data is outstanding, the sender determines that the ACK is a keepalive, meant to keep the line active, and it does nothing. If data is outstanding, the sender determines whether the ACK indicates that the receiver

has received some or none of the data. If the ACK indicates receipt of some data sent, the sender determines if new credit has been granted to allow it to send more data. When the ACK indicates receipt of none of the data sent and there is outstanding data, the sender interprets the ACK to be a repeatedly sent ACK. This condition indicates that some data was received out of order, forcing the receiver to remit the first ACK, and that a second data segment was received out of order, forcing the receiver to remit the second ACK. In most cases, the receiver would receive two segments out of order because one of the data segments had been dropped.

When a TCP sender detects a dropped data segment, it resends the segment. Then it adjusts its transmission rate to half of what is was before the drop was detected. This is the TCP back-off or slow-down behavior. Although this behavior is appropriately responsive to congestion, problems can arise when multiple TCP sessions are carried on concurrently with the same router and all TCP senders slow down transmission of packets at the same time.

How the Router Interacts with TCP

Note The sections How TCP Handles Traffic Loss, page 3 and How the Router Interacts with TCP, page 4 contain detailed information that you need not read in order to use WRED or to have a general sense of the capabilities of RED. If you want to understand why problems of global synchronization occur in response to congestion and how RED addresses them, read these sections.

To see how the router interacts with TCP, we will look at an example. In this example, on average, the router receives traffic from one particular TCP stream every other, every 10th, and every 100th or 200th message in the interface in MAE-EAST or FIX-WEST. A router can handle multiple concurrent TCP sessions. Because network flows are additive, there is a high probability that when traffic exceeds the Transmit Queue Limit (TQL) at all, it will vastly exceed the limit. However, there is also a high probability that the excessive traffic depth is temporary and that traffic will not stay excessively deep except at points where traffic flows merge or at edge routers.

If the router drops all traffic that exceeds the TQL, many TCP sessions will simultaneously go into slow start. Consequently, traffic temporarily slows down to the extreme and then all flows slow-start again; this activity creates a condition of global synchronization.

However, if the router drops no traffic, as is the case when queueing features such as fair queueing or priority queueing (PQ) are used, then the data is likely to be stored in main memory, drastically degrading router performance.

By directing one TCP session at a time to slow down, RED solves the problems described, allowing for full utilization of the bandwidth rather than utilization manifesting as crests and troughs of traffic.

About WRED

WRED combines the capabilities of the RED algorithm with the IP Precedence feature to provide for preferential traffic handling of higher priority packets. WRED can selectively discard lower priority traffic when the interface begins to get congested and provide differentiated performance characteristics for different classes of service.

You can configure WRED to ignore IP precedence when making drop decisions so that nonweighted RED behavior is achieved.

For interfaces configured to use the Resource Reservation Protocol (RSVP) feature, WRED chooses packets from other flows to drop rather than the RSVP flows. Also, IP Precedence governs which packets are dropped--traffic that is at a lower precedence has a higher drop rate and therefore is more likely to be throttled back.

WRED differs from other congestion avoidance techniques such as queueing strategies because it attempts to anticipate and avoid congestion rather than control congestion once it occurs.

Why Use WRED

WRED makes early detection of congestion possible and provides for multiple classes of traffic. It also protects against global synchronization. For these reasons, WRED is useful on any output interface where you expect congestion to occur.

However, WRED is usually used in the core routers of a network, rather than at the edge of the network. Edge routers assign IP precedences to packets as they enter the network. WRED uses these precedences to determine how to treat different types of traffic.

WRED provides separate thresholds and weights for different IP precedences, allowing you to provide different qualities of service in regard to packet dropping for different traffic types. Standard traffic may be dropped more frequently than premium traffic during periods of congestion.

WRED is also RSVP-aware, and it can provide the controlled-load QoS service of integrated service.

How It Works

By randomly dropping packets prior to periods of high congestion, WRED tells the packet source to decrease its transmission rate. If the packet source is using TCP, it will decrease its transmission rate until all the packets reach their destination, which indicates that the congestion is cleared.

WRED generally drops packets selectively based on IP precedence. Packets with a higher IP precedence are less likely to be dropped than packets with a lower precedence. Thus, the higher the priority of a packet, the higher the probability that the packet will be delivered.

WRED selectively drops packets when the output interface begins to show signs of congestion. By dropping some packets early rather than waiting until the queue is full, WRED avoids dropping large numbers of packets at once and minimizes the chances of global synchronization. Thus, WRED allows the transmission line to be used fully at all times.

In addition, WRED statistically drops more packets from large users than small. Therefore, traffic sources that generate the most traffic are more likely to be slowed down than traffic sources that generate little traffic.

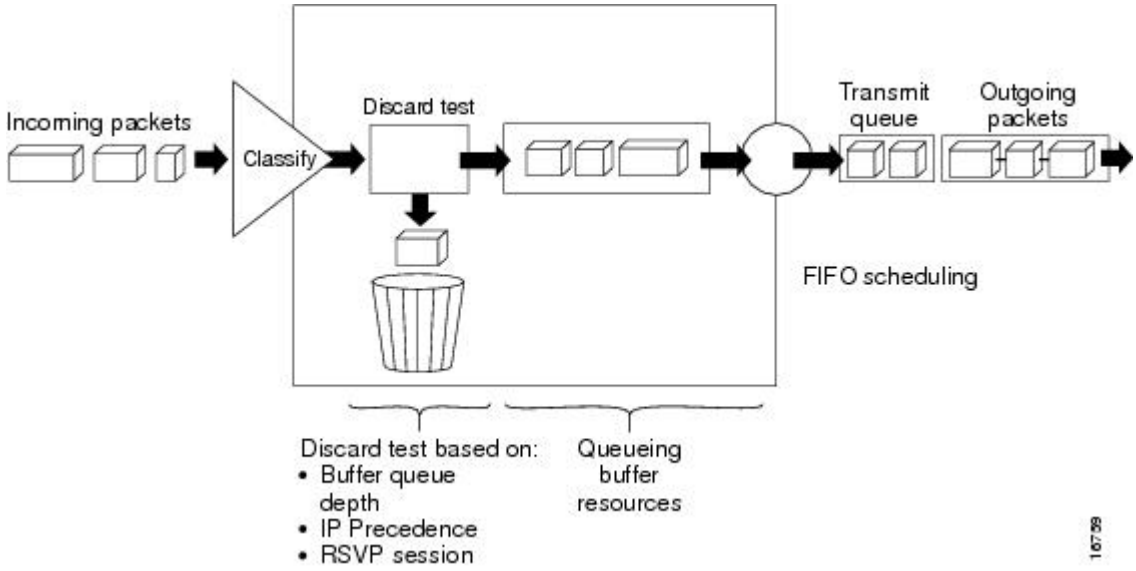
WRED helps to avoid the globalization problems. Global synchronization manifests when multiple TCP hosts reduce their transmission rates in response to packet dropping and then increase their transmission rates once again when the congestion is reduced.

WRED is only useful when the bulk of the traffic is TCP/IP traffic. With TCP, dropped packets indicate congestion, so the packet source will reduce its transmission rate. With other protocols, packet sources may not respond or may resend dropped packets at the same rate. Thus, dropping packets does not decrease congestion.

WRED treats non-IP traffic as precedence 0, the lowest precedence. Therefore, non-IP traffic, in general, is more likely to be dropped than IP traffic.

The figure below illustrates how WRED works.

Figure 2 Weighted Random Early Detection



Average Queue Size

The router automatically determines parameters to use in the WRED calculations. The average queue size is based on the previous average and the current size of the queue. The formula is:

```
average = (old_average * (1-2 -n)) + (current_queue_size * 2
-n
)
```

where *n* is the exponential weight factor, a user-configurable value. The default value of the exponential weight factor is 4. It is recommended to use only the default value for the exponential weight factor. Change this value from the default value only if you have determined that your scenario would benefit from using a different value.

For high values of *n*, the previous average becomes more important. A large factor smooths out the peaks and lows in queue length. The average queue size is unlikely to change very quickly, avoiding drastic swings in size. The WRED process will be slow to start dropping packets, but it may continue dropping packets for a time after the actual queue size has fallen below the minimum threshold. The slow-moving average will accommodate temporary bursts in traffic.

Note If the value of *n* gets too high, WRED will not react to congestion. Packets will be sent or dropped as if WRED were not in effect.

For low values of *n*, the average queue size closely tracks the current queue size. The resulting average may fluctuate with changes in the traffic levels. In this case, the WRED process responds quickly to long queues. Once the queue falls below the minimum threshold, the process will stop dropping packets.

If the value of *n* gets too low, WRED will overreact to temporary traffic bursts and drop traffic unnecessarily.

Configuring Weighted Random Early Detection

This module describes the tasks for configuring Weighted Random Early Detection (WRED) on a router.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

About Weighted Random Early Detection

Random Early Detection (RED) is a congestion avoidance mechanism that takes advantage of the congestion control mechanism of TCP. By randomly dropping packets prior to periods of high congestion, RED tells the packet source to decrease its transmission rate. WRED drops packets selectively based on IP precedence. Edge routers assign IP precedences to packets as they enter the network. (WRED is useful on any output interface where you expect to have congestion. However, WRED is usually used in the core routers of a network, rather than at the edge.) WRED uses these precedences to determine how it treats different types of traffic.

When a packet arrives, the following events occur:

1. The average queue size is calculated.
2. If the average is less than the minimum queue threshold, the arriving packet is queued.
3. If the average is between the minimum queue threshold for that type of traffic and the maximum threshold for the interface, the packet is either dropped or queued, depending on the packet drop probability for that type of traffic.
4. If the average queue size is greater than the maximum threshold, the packet is dropped.

Note WRED is useful with adaptive traffic such as TCP/IP. With TCP, dropped packets indicate congestion, so the packet source will reduce its transmission rate. With other protocols, packet sources may not respond or may resend dropped packets at the same rate. Thus, dropping packets does not decrease congestion. WRED treats non-IP traffic as precedence 0, the lowest precedence. Therefore, non-IP traffic is more likely to be dropped than IP traffic.

When you enable WRED with the **random-detect** interface configuration command, the parameters are set to their default values. The weight factor is 9. For all precedences, the mark probability denominator is 10, and maximum threshold is based on the output buffering capacity and the transmission speed for the interface.

The default minimum threshold depends on the precedence. The minimum threshold for IP Precedence 0 corresponds to half of the maximum threshold. The values for the remaining precedences fall between half the maximum threshold and the maximum threshold at evenly spaced intervals.

Note The default WRED parameter values are based on the best available data. We recommend that you do not change the parameters from their default values unless you have determined that your applications will benefit from the changed values.

How to Configure WRED

Enabling WRED

Command	Purpose
`Router(config-if)#` **random-detect**	Enables WRED.

Changing WRED Parameters

Command	Purpose
`Router(config-if)#` **random-detect exponential-weighting-constant** *exponent*	Configures the weight factor used in calculating the average queue length.

Command	Purpose
`Router(config-if)# random-detect precedence precedence min-threshold max-threshold mark-prob-denominator`	Configures parameters for packets with a specific IP Precedence. The minimum threshold for IP Precedence 0 corresponds to half the maximum threshold for the interface. Repeat this command for each precedence. To configure RED, rather than WRED, use the same parameters for each precedence.

Monitoring WRED

Command	Purpose
`Router# show queue interface-type interface-number`	Displays the header information of the packets inside a queue.
`Router# show queueing interface interface-number [vc [[vpi/] vci]]`	Displays the WRED statistics of a specific virtual circuit (VC) on an interface.
`Router# show queueing random-detect`	Displays the queueing configuration for WRED.
`Router# show interfaces [type slot | port-adapter | port]`	Displays WRED configuration on an interface.

WRED Configuration Examples

Example WRED Configuration

The following example enables WRED with default parameter values:

```
interface Serial5/0
 description to qos1-75a
 ip address 200.200.14.250 255.255.255.252
 random-detect
```

Use the **show interfaces** command output to verify the configuration. Notice that the "Queueing strategy" report lists "random early detection (RED)."

```
Router# show interfaces serial 5/0
Serial5/0 is up, line protocol is up
  Hardware is M4T
  Description: to qos1-75a
  Internet address is 200.200.14.250/30
  MTU 1500 bytes, BW 128 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 237/255
  Encapsulation HDLC, crc 16, loopback not set
```

```
  Keepalive not set
  Last input 00:00:15, output 00:00:00, output hang never
  Last clearing of "show interface" counters 00:05:08
  Input queue: 0/75/0 (size/max/drops); Total output drops: 1036
  Queueing strategy: random early detection(RED)
  5 minutes input rate 0 bits/sec, 2 packets/sec
  5 minutes output rate 119000 bits/sec, 126 packets/sec
     594 packets input, 37115 bytes, 0 no buffer
     Received 5 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     37525 packets output, 4428684 bytes, 0 underruns
     0 output errors, 0 collisions, 0 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Use the **show queue** command output to view the current contents of the interface queue. Notice that there is only a single queue into which packets from all IP precedences are placed after dropping has taken place. The output has been truncated to show only three of the five packets.

```
Router# show queue serial 5/0

Output queue for Serial5/0 is 5/0
Packet 1, linktype: ip, length: 118, flags: 0x288
  source: 190.1.3.4, destination: 190.1.2.2, id: 0x0001, ttl: 254,
  TOS: 128 prot: 17, source port 11111, destination port 22222
    data: 0x2B67 0x56CE 0x005E 0xE89A 0xCBA9 0x8765 0x4321
          0x0FED 0xCBA9 0x8765 0x4321 0x0FED 0xCBA9 0x8765
Packet 2, linktype: ip, length: 118, flags: 0x288
  source: 190.1.3.5, destination: 190.1.2.2, id: 0x0001, ttl: 254,
  TOS: 160 prot: 17, source port 11111, destination port 22222
    data: 0x2B67 0x56CE 0x005E 0xE89A 0xCBA9 0x8765 0x4321
          0x0FED 0xCBA9 0x8765 0x4321 0x0FED 0xCBA9 0x8765
Packet 3, linktype: ip, length: 118, flags: 0x280
  source: 190.1.3.6, destination: 190.1.2.2, id: 0x0001, ttl: 254,
  TOS: 192 prot: 17, source port 11111, destination port 22222
    data: 0x2B67 0x56CE 0x005E 0xE89A 0xCBA9 0x8765 0x4321
          0x0FED 0xCBA9 0x8765 0x4321 0x0FED 0xCBA9 0x8765
```

Use the **show queueing** command output to view the current settings for each of the precedences. Also notice that the default minimum thresholds are spaced evenly between half and the entire maximum threshold. Thresholds are specified in terms of packet count.

```
Router# show queueing
Current random-detect configuration:
     Serial5/0
       Queueing strategy:random early detection (WRED)
       Exp-weight-constant:9 (1/512)
       Mean queue depth:28

       Class   Random       Tail    Minimum    Maximum     Mark
                 drop       drop  threshold  threshold  probability
         0        330          0         20         40     1/10
         1        267          0         22         40     1/10
         2        217          0         24         40     1/10
         3        156          0         26         40     1/10
         4         61          0         28         40     1/10
         5          6          0         31         40     1/10
         6          0          0         33         40     1/10
         7          0          0         35         40     1/10
        rsvp        0          0         37         40     1/10
```

Example Parameter-Setting WRED

The following example enables WRED on the interface and specifies parameters for the different IP precedences:

```
interface Hssi0/0/0
 description 45Mbps to R1
```

```
ip address 10.200.14.250 255.255.255.252
random-detect
random-detect precedence 0 32 256 100
random-detect precedence 1 64 256 100
random-detect precedence 2 96 256 100
random-detect precedence 3 120 256 100
random-detect precedence 4 140 256 100
random-detect precedence 5 170 256 100
random-detect precedence 6 290 256 100
random-detect precedence 7 210 256 100
random-detect precedence rsvp 230 256 100
```

Feature Information for Configuring Weighted Random Early Detection

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 1 Feature Information for Configuring Weighted Random Early Detection

Feature Name	Releases	Feature Information
Class-Based Weighted Fair Queueing (CBWFQ) and Weighted Random Early Detection (WRED)	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 Series Routers. **Note** For information about CBWFQ, see the "Configuring Weighted Fair Queueing" module.
Random Early Detection (RED)	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 Series Routers.
Weighted Random Early Detection	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 Series Routers.
Weighted RED (WRED)	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 Series Routers.

and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

Byte-Based Weighted Random Early Detection

This module explains how to enable byte-based Weighted Random Early Detection (WRED), and set byte-based queue limits and WRED thresholds.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Restrictions for Byte-Based Weighted Random Early Detection

- WRED is only useful when the bulk of the traffic is TCP/IP traffic. With TCP, dropped packets indicate congestion, so the packet source will reduce its transmission rate. With other protocols, packet sources may not respond or may resend dropped packets at the same rate. Thus, dropping packets does not decrease congestion.
- You cannot configure byte-based WRED on a class in which the queue-limit is configured in milliseconds or packets.

Information About Byte-Based Weighted Random Early Detection

Changes in functionality of WRED

This feature extends the functionality of WRED. In previous releases, you specified the WRED actions based on the number of packets. With the byte-based WRED, you can specify WRED actions based on the number of bytes.

Changes in Queue Limit and WRED Thresholds

In Cisco IOS XE Release 2.4, the Cisco ASR 1000 Series Aggregation Services Routers support the addition of bytes as a unit of configuration for both queue limits and WRED thresholds. Therefore, as of this release, packet-based and byte-based limits are configurable, with some restrictions.

How to Configure Byte-Based Weighted Random Early Detection

Configuring Byte-Based WRED

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **class-map** *class-map-name*
4. **match ip precedence** ip-precedence-value
5. exit
6. **policy-map** *policy-name*
7. **class** *class-name*
8. **random-detect**
9. **random-detect precedence** *precedence min-threshold* **bytes** *max-threshold* **bytes** *mark-prob-denominator*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**class-map** *class-map-name* **Example:** `Router(config)# class-map c1`	Specifies the user-defined name of the traffic class.
Step 4	**match ip precedence** ip-precedence-value **Example:** `Router(config-cmap)# match ip precedence 1`	Specifies up to eight IP Precedence values used as match criteria.
Step 5	exit **Example:** `Router(config-cmap)# exit`	Exits from class-map configuration mode.
Step 6	**policy-map** *policy-name* **Example:** `Router(config)# policy-map p1`	Specifies the name of the traffic policy to configure.
Step 7	**class** *class-name* **Example:** `Router(config-pmap)# class c1`	Specifies the name of a predefined traffic class, which was configured with the **class-map** command, used to classify traffic to the traffic policy.

	Command or Action	Purpose
Step 8	**random-detect** **Example:** `Router(config-pmap-c)# random-detect`	Enables WRED.
Step 9	**random-detect precedence** *precedence min-threshold* **bytes** *max-threshold* **bytes** *mark-prob-denominator* **Example:** **Example:** `Router(config-pmap-c)# random-detect precedence 1 2000 bytes 3000 bytes 200`	Configures the parameters for bytes with a specific IP precedence.

Configuring the Queue Depth and WRED Thresholds

Be sure that your configuration satisfies the following conditions when configuring the queue depth and WRED thresholds:

- When configuring byte-based mode, the queue limit must be configured prior to the WRED threshold and before the service policy is applied.
- When setting the queue depth and WRED thresholds in an enhanced QoS policies aggregation configuration, the limits are supported only for the default class at a subinterface policy map and for any classes at the main interface policy map.

Note Consider the following restrictions when you configure the queue depth and WRED thresholds:

- Do not configure the queue limit unit before you configure a queueing feature for a traffic class.
- If you do not configure a queue limit, then the default mode is packets.
- When you configure WRED thresholds, the following restrictions apply:
  - The WRED threshold must use the same unit as the queue limit. For example, if the queue limit is in packets, then the WRED thresholds also must be in packets.
  - If you do not configure a queue limit in bytes, then the default mode is packets and you must also configure the WRED threshold in packets.
  - The queue limit size must be greater than the WRED threshold.
- The unit modes for either the queue limit or WRED thresholds cannot be changed dynamically after a service policy is applied.

>

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map-name*
4. **class** *class-name*
5. *qos-queueing-feature*
6. **queue-limit** *queue-limit-size* [**bytes** | **packets**]
7. **random-detect** [**dscp-based** | **prec-based**]
8. Do one of the following:
   - **random-detect dscp** dscp-value {*min-threshold max-threshold* | *min-threshold* **bytes** *max-threshold* **bytes**} [*max-probability-denominator*]
   -
   -
   - **random-detect precedence** precedence {*min-threshold max-threshold* | *min-threshold* **bytes** *max-threshold* **bytes**} max-probability-denominator

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map-name* **Example:** `Router(config)# policy-map main-interface`	Specifies the name of the traffic policy that you want to configure or modify and enters policy-map configuration mode.
Step 4	**class** *class-name* **Example:** `Router(config-pmap)# class AF1`	Specifies the name of the traffic class and enters policy-map class configuration mode.

Command or Action	Purpose
Step 5 *qos-queueing-feature* **Example:** `Router(config-pmap-c)# bandwidth remaining ratio 90`	Enters a QoS configuration command. Some of the queueing features that are currently supported are **bandwidth**, **priority**, and **shape**. **Note** Multiple QoS queueing commands can be entered at this step. However, due to dependencies between the queue limit and WRED thresholds, you should configure WRED after you configure the queue limit.
Step 6 **queue-limit** *queue-limit-size* [**bytes** | **packets**] **Example:** `Router(config-pmap-c)# queue-limit 547500 bytes`	Specifies the maximum number (from 1 to 8192000) of bytes or packets that the queue can hold for this class.
Step 7 **random-detect** [**dscp-based** | **prec-based**] **Example:** `Router(config-pmap-c)# random-detect dscp-based`	Enables WRED in either DSCP-based mode or precedence-based mode.
Step 8 Do one of the following: • **random-detect dscp** dscp-value {*min-threshold max-threshold* | *min-threshold* **bytes** *max-threshold* **bytes**} [*max-probability-denominator*] • • • **random-detect precedence** precedence {*min-threshold max-threshold* | *min-threshold* **bytes** *max-threshold* **bytes**} max-probability-denominator **Example:** `Router(config-pmap-c)# random-detect precedence 8 750000 bytes 750000 bytes`	Configures WRED parameters for a particular DSCP value or IP precedence. **Note** Use the *min-threshold max-threshold* arguments without the **bytes** keyword to configure packet-based thresholds, when the **queue-limit** unit is also packets (the default). Alternatively, use these arguments with the **bytes** keyword when the **queue-limit** unit is configured in bytes.

Examples

Correct Configuration

Invalid Configuration

Correct Configuration

Invalid Configuration

The following examples show both correct and invalid configurations to demonstrate some of the restrictions.

The following example shows the correct usage of setting the queue limit in bytes mode after the **bandwidth remaining ratio** queueing feature has been configured for a traffic class:

```
class AF1
    bandwidth remaining ratio 90
        queue-limit 750000 bytes
```

The following example shows an invalid configuration for the queue limit in bytes mode before the **bandwidth remaining ratio** queueing feature has been configured for a traffic class:

```
class AF1
    queue-limit 750000 bytes
        bandwidth remaining ratio 90
```

The following example shows the correct usage of setting the queue limit in bytes mode after the **bandwidth remaining ratio** queueing feature has been configured for a traffic class, followed by the setting of the thresholds for WRED in compatible byte mode:

```
class AF1
    bandwidth remaining ratio 90
        queue-limit 750000 bytes
            random-detect dscp-based
                random-detect dscp 8 750000 bytes 750000 bytes
```

This example shows an invalid configuration of the WRED threshold in bytes without any queue limit configuration, which therefore defaults to a packet-based queue depth. Therefore, the WRED threshold must also be in packets:

```
class AF1
    bandwidth remaining ratio 90
    random-detect dscp-based
    random-detect dscp 8 750000 bytes 750000 bytes
```

Changing the Queue Depth and WRED Threshold Unit Modes

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **interface** *type number*
4. **no service-policy output** *policy-map-name*
5. **exit**
6. **policy-map** *policy-map-name*
7. **class** *class-name*
8. **queue-limit** *queue-limit-size* [**bytes** | **packets**]
9. Do one of the following:
   - **no random-detect dscp** *dscp-value* {*min-threshold max-threshold* | *min-threshold* **bytes** *max-threshold* **bytes**} [*max-probability-denominator*]
   -
   -
   - **no random-detect precedence** *precedence* {*min-threshold max-threshold* | *min-threshold* **bytes** *max-threshold* **bytes**} *max-probability-denominator*
10. Do one of the following:
    - **random-detect dscp** *dscp-value* {*min-threshold max-threshold* | *min-threshold* **bytes** *max-threshold* **bytes**} [*max-probability-denominator*]
    -
    -
    - **random-detect precedence** *precedence* {*min-threshold max-threshold*| *min-threshold* **bytes** *max-threshold* **bytes**} *max-probability-denominator*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 3	**interface** *type number* **Example:** `Router(config)# policy-map main-interface`	Specifies the interface where you want to remove a service policy, and enters interface configuration mode.
Step 4	**no service-policy output** *policy-map-name* **Example:** `Router(config-if)# no service-policy output main-interface-policy`	Removes a service policy applied to the specified interface.
Step 5	**exit** **Example:** `Router(config-if)# exit`	Exits interface configuration mode and returns you to global configuration mode.
Step 6	**policy-map** *policy-map-name* **Example:** `Router(config)# policy-map main-interface-policy`	Specifies the name of the Traffic policy that you want to modify and enters policy-map configuration mode.
Step 7	**class** *class-name* **Example:** `Router(config-pmap)# class AF1`	Specifies the name of the traffic class and enters policy-map class configuration mode.
Step 8	**queue-limit** *queue-limit-size* [**bytes** | **packets**] **Example:** `Router(config-pmap-c)# queue-limit 5000 packets`	Specifies the maximum number (from 1 to 8192000) of bytes or packets that the queue can hold for this class.

	Command or Action	Purpose
Step 9	Do one of the following: • **no random-detect dscp** *dscp-value* {*min-threshold max-threshold* | *min-threshold* **bytes** *max-threshold* **bytes**} [*max-probability-denominator*] • • • **no random-detect precedence** *precedence* {*min-threshold max-threshold* | *min-threshold* **bytes** *max-threshold* **bytes**} *max-probability-denominator* **Example:** `Router(config-pmap-c)# no random-detect dscp 8 750000 bytes 750000 bytes`	Removes the previously configured WRED parameters for a particular DSCP value or IP precedence.
Step 10	Do one of the following: • **random-detect dscp** *dscp-value* {*min-threshold max-threshold* | *min-threshold* **bytes** *max-threshold* **bytes**} [*max-probability-denominator*] • • • **random-detect precedence** *precedence* {*min-threshold max-threshold*| *min-threshold* **bytes** *max-threshold* **bytes**} *max-probability-denominator* **Example:** `Router(config-pmap-c)# random-detect dscp 8 4000 4000`	Configures WRED parameters for a particular DSCP value or IP precedence. **Note** Use the *min-threshold max-threshold* arguments without the **bytes** keyword to configure packet-based thresholds, when the queue-limit unit is also packets (the default). Alternatively, use these arguments with the **bytes** keyword when the queue-limit unit is configured in bytes.

Examples

The following example shows how to change the queue depth and WRED thresholds to packet-based values once a service policy has been applied to an interface:

```
interface GigabitEthernet1/2/0
no service-policy output main-interface-policy
end
policy-map main-interface-policy
class AF1
    queue-limit 5000 packets
    no random-detect dscp 8 750000 bytes 750000 bytes
    random-detect dscp 8 4000 4000
```

Verifying the Configuration for Byte-Based WRED

SUMMARY STEPS

1. **show policy-map**
2. The **show policy-map interface** command shows output for an interface that is configured for byte-based WRED.

DETAILED STEPS

Step 1 **show policy-map**

The **show policy-map** command shows the output for a service policy called pol1 that is configured for byte-based WRED.

Example:

```
Router# show policy-map
Policy Map pol1
  Class class c1
Bandwidth 10 (%)
exponential weight 9
        class   min-threshold(bytes)   max-threshold(bytes)  mark-probability
        ----------------------------------------------------------------------

        0       -                     -                     1/10
        1       20000                 30000                 1/10
        2       -                     -                     1/10
        3       -                     -                     1/10
        4       -                     -                     1/10
        5       -                     -                     1/10
        6       -                     -                     1/10
        7       -                     -                     1/10
        rsvp    -                     -                     1/10
```

Step 2 The **show policy-map interface** command shows output for an interface that is configured for byte-based WRED.

Example:

```
Router# show policy-map interface
serial3/1
Service-policy output: pol
Class-map: silver (match-all)
366 packets, 87840 bytes
30 second offered rate 15000 bps, drop rate 300 bps
Match: ip precedence 1
Queueing
Output Queue: Conversation 266
Bandwidth 10 (%)
(pkts matched/bytes matched) 363/87120
depth/total drops/no-buffer drops) 147/38/0
exponential weight: 9
mean queue depth: 25920
class     Transmitted       Random drop      Tail drop     Minimum Maximum Mark
          pkts/bytes        pkts/bytes       pkts/bytes    thresh  thresh  prob
                                                           (bytes)  (bytes)
0             0/0               0/0              0/0      20000    40000  1/10
1           328/78720         38/9120            0/0      22000    40000  1/10
2             0/0               0/0              0/0      24000    40000  1/10
```

```
3             0/0              0/0              0/0      26000    40000  1/10
4             0/0              0/0              0/0      28000    40000  1/10
```

Configuration Examples for Byte-Based Weighted Random Early Detection

Example Configuring Byte-Based WRED

The following example shows a service policy called wred-policy that sets up byte-based WRED for a class called prec2 and for the default class. The policy is then applied to Fast Ethernet interface 0/0/1.

```
policy wred-policy
 class prec2
   bandwidth 1000
   random-detect
   random-detect precedence 2 100 bytes 200 bytes 10
class class-default
   random-detect
   random-detect precedence 4 150 bytes 300 bytes 15
   random-detect precedence 6 200 bytes 400 bytes 5
interface fastethernet0/0/1
  service-policy output wred-policy
```

The following example shows the byte-based WRED results for the service policy attached to Ethernet interface 0/0/1.

```
Router# show policy-map interface
 Ethernet0/0/1
Service-policy output: wred-policy (1177)
Class-map: prec2 (match-all) (1178/10)
0 packets, 0 bytes
5 minute offered rate 0 bps, drop rate 0 bps
Match: ip precedence 2  (1179)
Queueing
queue limit 62500 bytes
(queue depth/total drops/no-buffer drops) 0/0/0
(pkts queued/bytes queued) 0/0
bandwidth 1000 (kbps)
Exp-weight-constant: 9 (1/512)
Mean queue depth: 0 bytes
class     Transmitted       Random drop      Tail drop Minimum        Maximum     Mark
          pkts/bytes        pkts/bytes       pkts/bytes thresh         thresh     prob
                                                        bytes          bytes
0             0/0               0/0              0/0     15625          31250     1/10
1             0/0               0/0              0/0     17578          31250     1/10
2             0/0               0/0              0/0       100            200     1/10
3             0/0               0/0              0/0     21484          31250     1/10
4             0/0               0/0              0/0     23437          31250     1/10
5             0/0               0/0              0/0     25390          31250     1/10
6             0/0               0/0              0/0     27343          31250     1/10
7             0/0               0/0              0/0     29296          31250     1/10
Class-map: class-default (match-any) (1182/0)
0 packets, 0 bytes
5 minute offered rate 0 bps, drop rate 0 bps
Match: any  (1183)
0 packets, 0 bytes
```

```
5 minute rate 0 bps
queue limit 562500 bytes
(queue depth/total drops/no-buffer drops) 0/0/0
(pkts queued/bytes queued) 0/0
Exp-weight-constant: 9 (1/512)
Mean queue depth: 0 bytes
class     Transmitted       Random drop      Tail drop Minimum        Maximum     Mark
          pkts/bytes        pkts/bytes       pkts/bytes thresh         thresh     prob
                                                        bytes         bytes
0             0/0               0/0              0/0     140625        281250     1/10
1             0/0               0/0              0/0     158203        281250     1/10
2             0/0               0/0              0/0     175781        281250     1/10
3             0/0               0/0              0/0     193359        281250     1/10
4             0/0               0/0              0/0        150           300     1/15
5             0/0               0/0              0/0     228515        281250     1/10
6             0/0               0/0              0/0        200           400     1/5
7             0/0               0/0              0/0     263671        281250     1/10
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
QoS Commands	*Cisco IOS Quality of Service Solutions Command Reference*
Modular QoS CLI	Modular Quality of Service Command-Line Interface module

Standards

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported, and support for existing MIBs has not been modified.	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
No new or modified RFCs are supported, and support for existing RFCs has not been modified.	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Byte-Based Weighted Random Early Detection

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 2 Feature Information for Byte-Based Weighted Random Early Detection

Feature Name	Releases	Feature Information
Byte-Based Weighted Random Early Detection	Cisco IOS XE Release 2.4	The Byte-Based Weighted Random Early Detection feature extends the functionality of WRED. In previous releases, you specified the WRED actions based on the number of packets. With the byte-based WRED, you can specify WRED actions based on the number of bytes. This feature was introduced on Cisco ASR 1000 Series Routers. The following commands were introduced or modified: **random-detect**, **random-detect precedence**, **show policy-map, show policy-map interface**.

WRED Explicit Congestion Notification

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the Feature Information Table at the end of this document.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for WRED Explicit Congestion Notification

ECN must be configured through the Modular Quality of Service Command-Line Interface (MQC).

Information About WRED Explicit Congestion Notification

WRED Explicit Congestion Notification Feature Overview

Currently, the congestion control and avoidance algorithms for Transmission Control Protocol (TCP) are based on the idea that packet loss is an appropriate indication of congestion on networks transmitting data using the best-effort service model. When a network uses the best-effort service model, the network

delivers data if it can, without any assurance of reliability, delay bounds, or throughput. However, these algorithms and the best-effort service model are not suited to applications that are sensitive to delay or packet loss (for instance, interactive traffic including Telnet, web-browsing, and transfer of audio and video data). Weighted Random Early Detection (WRED), and by extension, Explicit Congestion Notification (ECN), helps to solve this problem.

RFC 3168, *The Addition of Explicit Congestion Notification (ECN) to IP*, states that with the addition of active queue management (for example, WRED) to the Internet infrastructure, routers are no longer limited to packet loss as an indication of congestion.

How WRED Works

WRED makes early detection of congestion possible and provides a means for handling multiple classes of traffic. WRED can selectively discard lower priority traffic when the router begins to experience congestion and provide differentiated performance characteristics for different classes of service. It also protects against global synchronization. Global synchronization occurs as waves of congestion crest, only to be followed by periods of time during which the transmission link is not used to capacity. For these reasons, WRED is useful on any output interface or router where congestion is expected to occur.

WRED is implemented at the core routers of a network. Edge routers assign IP precedences to packets as the packets enter the network. With WRED, core routers then use these precedences to determine how to treat different types of traffic. WRED provides separate thresholds and weights for different IP precedences, enabling the network to provide different qualities of service, in regard to packet dropping, for different types of traffic. Standard traffic may be dropped more frequently than premium traffic during periods of congestion.

ECN Extends WRED Functionality

WRED drops packets, based on the average queue length exceeding a specific threshold value, to indicate congestion. ECN is an extension to WRED in that ECN marks packets instead of dropping them when the average queue length exceeds a specific threshold value. When configured with the WRED -- Explicit Congestion Notification feature, routers and end hosts would use this marking as a signal that the network is congested and slow down sending packets.

As stated in RFC 3168, *The Addition of Explicit Congestion Notification (ECN) to IP*,implementing ECN requires an ECN-specific field that has two bits--the ECN-capable Transport (ECT) bit and the CE (Congestion Experienced) bit--in the IP header. The ECT bit and the CE bit can be used to make four ECN field combinations of 00 to 11. The first number is the ECT bit and the second number is the CE bit. The table below lists each of the ECT and CE bit combination settings in the ECN field and what the combinations indicate.

Table 3 ECN Bit Setting

ECT Bit	CE Bit	Combination Indicates
0	0	Not ECN-capable
0	1	Endpoints of the transport protocol are ECN-capable
1	0	Endpoints of the transport protocol are ECN-capable

ECT Bit	CE Bit	Combination Indicates
1	1	Congestion experienced

The ECN field combination 00 indicates that a packet is not using ECN.

The ECN field combinations 01 and 10--called ECT(1) and ECT(0), respectively--are set by the data sender to indicate that the endpoints of the transport protocol are ECN-capable. Routers treat these two field combinations identically. Data senders can use either one or both of these two combinations. For more information about these two field combinations, and the implications of using one over the other, refer to RFC 3168, *The Addition of Explicit Congestion Notification (ECN) to IP*.

The ECN field combination 11 indicates congestion to the endpoints. Packets arriving a full queue of a router will be dropped.

- How Packets Are Treated When ECN Is Enabled, page 33

How Packets Are Treated When ECN Is Enabled

- If the number of packets in the queue is below the minimum threshold, packets are transmitted. This happens whether or not ECN is enabled, and this treatment is identical to the treatment a packet receives when WRED only is being used on the network.
- If the number of packets in the queue is between the minimum threshold and the maximum threshold, one of the following three scenarios can occur:
  - If the ECN field on the packet indicates that the endpoints are ECN-capable (that is, the ECT bit is set to 1 and the CE bit is set to 0, or the ECT bit is set to 0 and the CE bit is set to 1)--and the WRED algorithm determines that the packet should have been dropped based on the drop probability--the ECT and CE bits for the packet are changed to 1, and the packet is transmitted. This happens because ECN is enabled and the packet gets marked instead of dropped.
  - If the ECN field on the packet indicates that neither endpoint is ECN-capable (that is, the ECT bit is set to 0 and the CE bit is set to 0), the packet may be dropped based on the WRED drop probability. This is the identical treatment that a packet receives when WRED is enabled without ECN configured on the router.
  - If the ECN field on the packet indicates that the network is experiencing congestion (that is, both the ECT bit and the CE bit are set to 1), the packet is transmitted. No further marking is required.
- If the number of packets in the queue is above the minimum threshold, packets are dropped based on the drop probability. This is the identical treatment a packet receives when WRED is enabled without ECN configured on the router.

Benefits of WRED Explicit Congestion Notification

Improved Method for Congestion Avoidance

This feature provides an improved method for congestion avoidance by allowing the network to mark packets for transmission later, rather than dropping them from the queue. Marking the packets for transmission later accommodates applications that are sensitive to delay or packet loss and provides improved throughput and application performance.

Enhanced Queue Management

Currently, dropped packets indicate that a queue is full and that the network is experiencing congestion. When a network experiences congestion, this feature allows networks to mark the IP header of a packet

with a CE bit. This marking, in turn, triggers the appropriate congestion avoidance mechanism and allows the network to better manage the data queues. With this feature, ECN-capable routers and end hosts can respond to congestion before a queue overflows and packets are dropped, providing enhanced queue management.

How to Configure WRED Explicit Congestion Notification

Configuring Explicit Congestion Notification

To configure ECN, complete the following steps.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **policy-map** *policy-map-name*
4. **class** {*class-name*| **class-default**}
5. **bandwidth** {*bandwidth-kbps* | **percent** *percent*
6. **random-detect**
7. **random-detect ecn**
8. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**policy-map** *policy-map-name* **Example:** `Router(config)# policy-map policy1`	Creates or modifies a policy map that can be attached to one or more interfaces to specify a service policy. Enters QoS policy-map configuration mode. • Enter the name of the policy map.

	Command or Action	Purpose
Step 4	**class** {*class-name*| **class-default**} **Example:** `Router(config-pmap)# class class-default`	Specifies the name of the class whose policy you want to create or change or specifies the default class (commonly known as the class-default class) before you configure its policy. Enters policy-map-class configuration mode. • Enter the name of the class or enter the **class-default**keyword.

Step 5	**bandwidth** {*bandwidth-kbps* | **percent** *percent*
	
	Example:
	`Router(config-pmap-c)#`
	`bandwidth percent 35`
		Specifies or modifies the bandwidth (either in kbps or a percentage) allocated for a class belonging to a policy map.
		• Enter the bandwidth in kilobytes per second or enter the bandwidth percentage.

Step 6	**random-detect**
	
	Example:
	`Router(config-pmap-c)#`
	`random-detect`
		Enables WRED or distributed WRED (dWRED).

Step 7	**random-detect ecn**
	
	Example:
	`Router(config-pmap-c)#`
	`random-detect ecn`
		Enables ECN.

Step 8	**end**
	
	Example:
	`Router(config-pmap-c)#`
	`end`
		(Optional) Exits policy-map class configuration mode.

Verifying the Explicit Congestion Notification Configuration

SUMMARY STEPS

1. **enable**
2. **show policy-map**
3. **show policy-map interface**
4. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**show policy-map** **Example:** `Router# show policy-map`	If ECN is enabled, displays ECN marking information for a specified policy map.
Step 3	**show policy-map interface** **Example:** `Router# show policy-map interface`	If ECN is enabled, displays ECN marking information for a specified interface.

Step 4	**end**
	
	Example:
	`Router#`
	`end`
		(Optional) Exits privileged EXEC mode.

Configuration Examples for WRED Explicit Congestion Notification

Example Enabling ECN

The following example enables ECN in the policy map called pol1:

```
Router(config)# policy-map pol1
 Router(config-pmap)# class class-default
 Router(config-pmap-c)# bandwidth per 70
 Router(config-pmap-c)# random-detect
 Router(config-pmap-c)# random-detect ecn
```

Example Verifying the ECN Configuration

The following is sample output from the **show policy-map** command. The words "explicit congestion notification" (along with the ECN marking information) included in the output indicate that ECN has been enabled.

```
Router# show policy-map
   Policy Map pol1
     Class class-default
       Weighted Fair Queueing
             Bandwidth 70 (%)
             exponential weight 9
             explicit congestion notification
             class    min-threshold    max-threshold    mark-probability
             ------------------------------------------------------------
             ------------------------------------------------------------

             0        -                -                1/10
             1        -                -                1/10
             2        -                -                1/10
             3        -                -                1/10
             4        -                -                1/10
             5        -                -                1/10
             6        -                -                1/10
             7        -                -                1/10
             rsvp     -                -                1/10
```

The following is sample output from the **show policy-map interface**command. The words "explicit congestion notification" included in the output indicate that ECN has been enabled.

```
Router# show policy-map interface
 Serial4/1
 Serial4/1
  Service-policy output:policy_ecn
        Class-map:prec1 (match-all)
          1000 packets, 125000 bytes
          30 second offered rate 14000 bps, drop rate 5000 bps
          Match:ip precedence 1
          Weighted Fair Queueing
            Output Queue:Conversation 42
            Bandwidth 20 (%)
            Bandwidth 100 (kbps)
            (pkts matched/bytes matched) 989/123625
        (depth/total drops/no-buffer drops) 0/455/0
             exponential weight:9
             explicit congestion notification
             mean queue depth:0
    class   Transmitted  Random drop  Tail drop   Minimum     Maximum     Mark
            pkts/bytes   pkts/bytes    pkts/bytes threshold   threshold   probability
      0       0/0          0/0          0/0          20          40        1/10
      1     545/68125      0/0          0/0          22          40        1/10
      2       0/0          0/0          0/0          24          40        1/10
      3       0/0          0/0          0/0          26          40        1/10
      4       0/0          0/0          0/0          28          40        1/10
      5       0/0          0/0          0/0          30          40        1/10
      6       0/0          0/0          0/0          32          40        1/10
      7       0/0          0/0          0/0          34          40        1/10
    rsvp      0/0          0/0          0/0          36          40        1/10
    class   ECN Mark
           pkts/bytes
      0     0/0
      1    43/5375
      2     0/0
      3     0/0
      4     0/0
      5     0/0
      6     0/0
      7     0/0
    rsvp    0/0
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
QoS commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Cisco IOS Quality of Service Solutions Command Reference*
MQC	"Applying QoS Features Using the MQC" module
Congestion avoidance concepts	"Congestion Avoidance Overview" module

Standards

Standard	Title
No new or modified standards are supported, and support for existing standards has not been modified.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported, and support for existing MIBs has not been modified.	To locate and download MIBs for selected platforms, Cisco IOS releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

RFCs

RFC	Title
RFC 2309	*Internet Performance Recommendation*
RFC 2884	*Performance Evaluation of Explicit Congestion Notification (ECN) in IP Networks*
RFC 3168	*The Addition of Explicit Congestion Notification (ECN) to IP*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for WRED Explicit Congestion Notification

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 4 Feature Information for WRED Explicit Congestion Notification

Feature Name	Software Releases	Feature Configuration Information
WRED Explicit Congestion Notification	Cisco IOS XE Release 2.1	Currently, the congestion control and avoidance algorithms for Transmission Control Protocol (TCP) are based on the idea that packet loss is an appropriate indication of congestion on networks transmitting data using the best-effort service model. When a network uses the best-effort service model, the network delivers data if it can, without any assurance of reliability, delay bounds, or throughput. However, these algorithms and the best-effort service model are not suited to applications that are sensitive to delay or packet loss (for instance, interactive traffic including Telnet, web-browsing, and transfer of audio and video data). Weighted Random Early Detection (WRED), and by extension, Explicit Congestion Notification (ECN), helps to solve this problem. The following commands were introduced or modified: **random-detect ecn**, **show policy-map**, **show policy-map interface**.