

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 QoS includes an implementation of RED that, when configured, controls when the router drops packets. If you do not configure Weighted Random Early Detection (WRED), the router uses the cruder default packet drop mechanism called tail drop.

Note For explanation of network congestion, see the chapter “Introduction: Quality of Service Overview.”

This chapter gives a brief description of the kinds of congestion avoidance features provided by the Cisco IOS QoS features. It discusses the following features:

- Tail Drop. This is the default congestion avoidance behavior when WRED is not configured.
- Weighted Random Early Detection. WRED and VIP-Distributed WRED—both of which are Cisco’s implementations of RED—combine the capabilities of the RED algorithm with IP Precedence.

Tail Drop

Tail drop treats all traffic equally and does not differentiate between classes of service. Queues fill during periods of congestion. When the output queue is full and tail drop is in effect, packets are dropped until the congestion is eliminated and the queue is no longer full.

Weighted Random Early Detection

This section gives a brief introduction to RED concepts and addresses WRED, Cisco’s implementation of RED for standard IOS platforms.

WRED avoids the globalization problems that occur when tail drop is used as the congestion avoidance mechanism on the router. 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 Transmission Control Protocol (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, then increase their transmission rates once again when the congestion is reduced.

This section comprises:

- About Random Early Detection
- About Weighted Random Early Detection
- VIP-Distributed Weighted Random Early Detection

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 which 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 RED’s traffic-drop behavior 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 RED’s usefulness 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 are 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 TCP’s congestion control mechanism. 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 explication of how Cisco’s 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” later in this chapter.

Packet Drop Probability

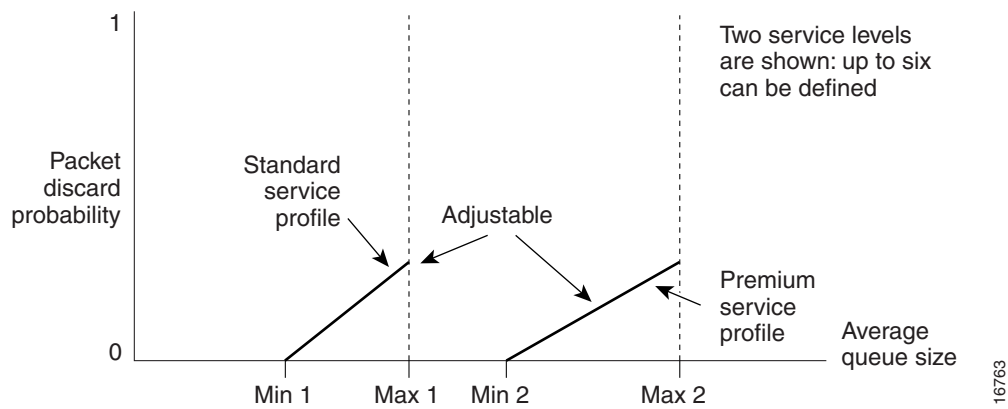
The packet drop probability is based on the minimum threshold, maximum threshold, and mark probability denominator.

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. Figure 8 summarizes the packet drop probability.

Figure 8 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. If the difference 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” and “How the Router Interacts with TCP” contain detailed information that you need not read in order to use WRED or to have a general sense of RED’s capabilities. If you want to understand why problems of global synchronization occur in response to congestion when tail drop is used by default and how RED addresses them, read these or greater sections.

When the recipient of TCP traffic—called the receiver—receives a data segment, it checks that data segment’s four octet (32-bit) sequence number 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 transmits an acknowledgment (ACK) packet to the sender or it schedules an ACK to be transmitted 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 piggyback 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 retransmit the lost data segment.

When the sender receives an ACK, it makes these determinations: It determines if any data is outstanding. If not, 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 acknowledges receipt of some data sent, the sender determines if new credit has been granted to allow it to send more data. When the ACK acknowledges 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 retransmit the first ACK, and that a second data segment was received out of order, forcing the receiver to retransmit 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 retransmits the segment. Then it adjusts its transmission rate so that it is half of what it 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” and “How the Router Interacts with TCP” contain detailed information that you need not read in order to use WRED or to have a general sense of RED’s capabilities. If you want to understand why problems of global synchronization occur in response to congestion when tail drop is used by default and how RED addresses them, read these sections.

For 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, as is done when tail drop is used by default, many TCP sessions will simultaneously go into slow start. Consequently, traffic temporarily slows down to the extreme and then all flows slow-start up 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 (FQ) or custom queueing (CQ) 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 Weighted Random Early Detection

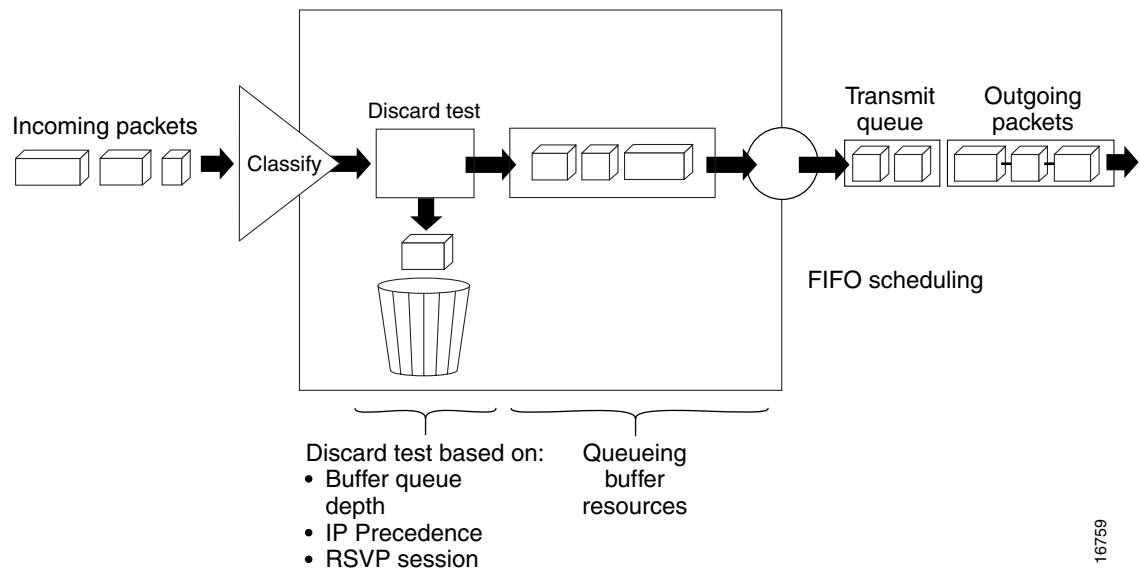
WRED combines the capabilities of the RED algorithm with IP Precedence to provide for preferential traffic handling of higher priority packets. WRED can selectively discard lower priority traffic when the interface begins to get congested and provide differentiated performance characteristics for different 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), 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. WRED is available on the Cisco 7200 series RSP processors. Figure 9 illustrates how WRED works.

Figure 9 Weighted Random Early Detection



16759

Why Use Weighted Random Early Detection?

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 the network's edge. 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 integrated services controlled-load QoS 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 reduces the chances of tail drop by selectively dropping 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 avoids the globalization problems that occur when tail drop is used as the congestion avoidance mechanism. Global synchronization manifests when multiple TCP hosts reduce their transmission rates in response to packet dropping, 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.

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

$$\text{average} = (\text{old_average} * (1 - 2^{-n})) + (\text{current_queue_size} * 2^{-n})$$

where n is the exponential weight factor, a user-configurable value.

Cisco recommends using the default value for the exponential weight factor. Change this value from the default only if you have determined that your situation 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 transmitted 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.

VIP-Distributed Weighted Random Early Detection

VIP-Distributed WRED is an implementation of WRED for the Versatile Interface Processor (VIP). VIP-Distributed WRED provides the complete set of functions for the VIP that WRED provides on standard IOS platforms. VIP-Distributed WRED is supported on the following routers with a VIP2-40 or greater interface processor:

- Cisco 7000 series with RSP7000
- Cisco 7500 series

Note As its name implies, the VIP-Distributed WRED feature uses the VIP rather than the Route Switch Processor (RSP) to perform queueing, which is why it requires a Cisco 7500 series router or Cisco 7000 series router with RSP7000.

A VIP2-50 interface processor is strongly recommended when the aggregate line rate of the port adapters on the VIP is greater than DS3. A VIP2-50 interface processor is required for OC-3 rates.

VIP-Distributed WRED is configured the same way as WRED. If you enable WRED on a suitable VIP interface, such as a VIP2-40 or greater with at least 2 MB of SRAM, VIP-Distributed WRED will be enabled instead.

In order to use VIP-Distributed WRED, Distributed Cisco Express Forwarding (DCEF) switching must be enabled on the interface. For information about DCEF, refer to the *Cisco IOS Switching Services Configuration Guide* and the *Cisco IOS Switching Services Command Reference*.

You can configure both VIP-Distributed WRED and Distributed weighted fair queueing (DWFQ) on the same interface, but you cannot configure VIP-Distributed WRED on an interface for which RSP-based CQ, priority queueing (PQ), or WFQ are configured.

Why Use VIP-Distributed Weighted Random Early Detection?

VIP-Distributed WRED provides faster performance than does RSP-based WRED. You should run WRED on the VIP if you want to achieve very high speed on the Cisco 7500 series platform—for example, you can achieve speed at the OC-3 rates by running WRED on a VIP2-50 interface processor.

Additionally, the same reasons you would use WRED on standard IOS platforms apply to using VIP-Distributed WRED on the VIP. (See the section “Why Use Weighted Random Early Detection?” earlier in this chapter.) For instance, when WRED or VIP Distributed-WRED are not configured, tail drop is enacted during periods of congestion. Enabling VIP-Distributed WRED on the VIP obviates the global synchronization problems that result when tail drop is used to avoid congestion.

Restrictions

The following restrictions apply to VIP-Distributed WRED:

- Interface-based DWRED cannot be configured on a subinterface. (A subinterface is one of a number of virtual interfaces on a single physical interface.)
- VIP-Distributed WRED is not supported on Fast EtherChannel and tunnel interfaces.
- RSVP is not supported on VIP-Distributed WRED.

