Compare Traffic Policing and Traffic Shaping to Limit Bandwidth

Contents

Introduction

Prerequisites

- Requirements
- Components Used

Background Information

Policing Versus Shaping

Selection Criteria

Token Refresh Rate

Traffic Shaping

Traffic Policing

Minimum Versus Maximum Bandwidth Controls

Related Information

Introduction

This document describes the functional differences between traffic shaping and traffic policing, both of which limit the output rate.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

This document is not restricted to specific software and hardware versions.

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, ensure that you understand the potential impact of any command.

Background Information

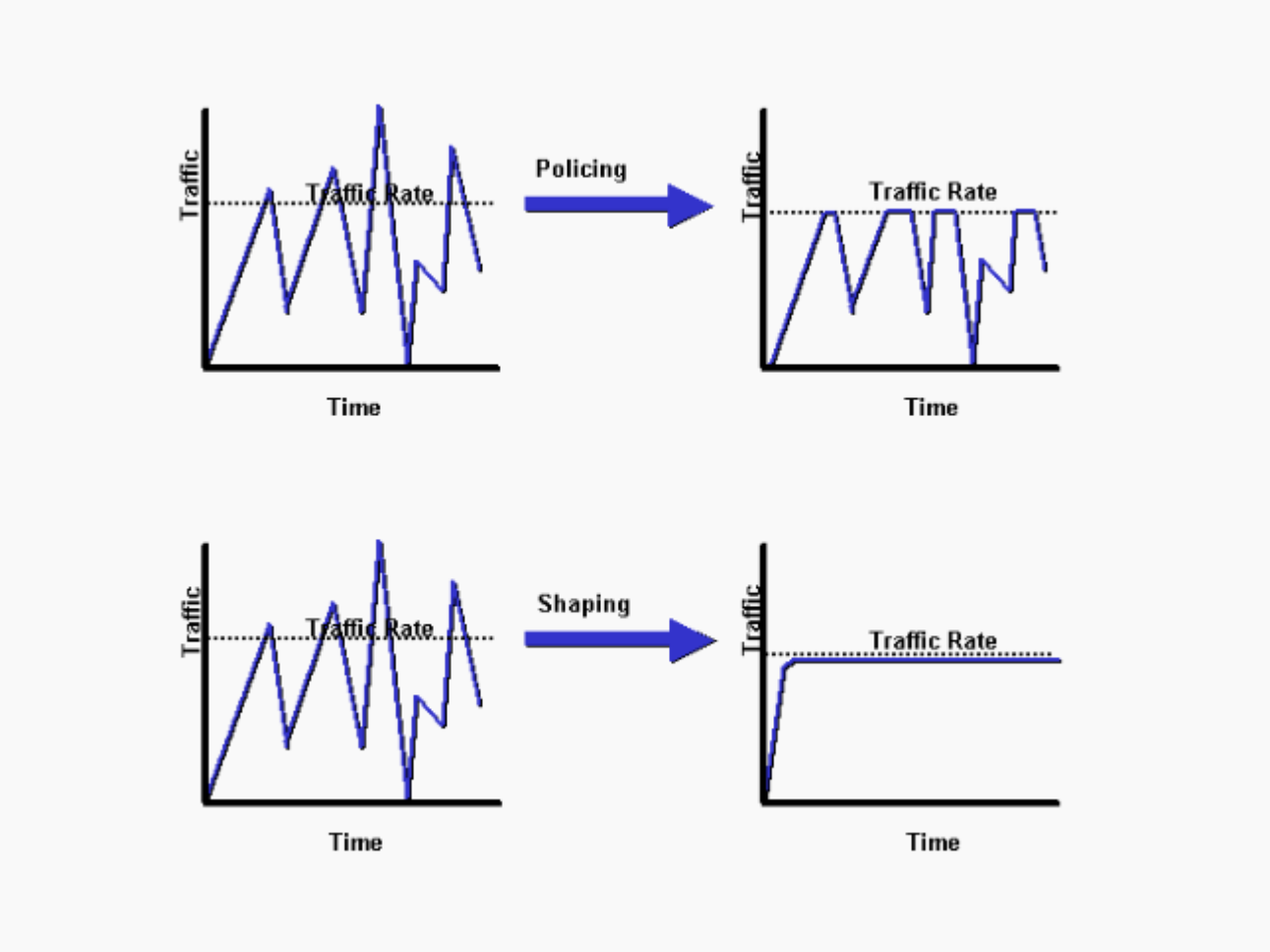
This document clarifies the functional differences between traffic shaping and policing. Both functionally limit the traffic output rate. Both mechanisms use a token bucket as a traffic meter to measure the packet rate. For more information on token buckets, see What Is a Token Bucket

The concepts in this document remain relevant for understanding Cisco QoS designs. Traffic shaping and traffic policing continue to be used to enforce maximum traffic rates and manage congestion. For new deployments, Cisco recommends using Modular QoS Command-Line (MQC) class-based shaping and class-based policing when supported by the platform. Legacy mechanisms, such as Generic Traffic Shaping (GTS), Frame Relay Traffic Shaping (FRTS), Committed Access Rate (CAR), and the rate-limit command, are included for historical context or to support platform specific discussions.

Policing Versus Shaping

Traffic policing propagates bursts. When the traffic rate reaches the configured maximum rate, excess traffic is dropped (or remarked). The result is an output rate that appears as a saw-tooth with crests and troughs. In contrast to policing, traffic shaping retains excess packets in a queue and then schedules the excess for later transmission over increments of time. The result of traffic shaping is a smoothed packet output rate.

The next diagram illustrates the key differences between the two traffic options.



Policing VS Shaping

Shaping implies the existence of a queue and of sufficient memory to buffer delayed packets, while policing does not. Queues are an outbound concept; packets that leave an interface get queued and can be shaped. Only policing can be applied to inbound traffic on an interface. Ensure that you have sufficient memory when you enable shaping. In addition, shaping requires a function that schedules for later transmission of any delayed packets. This schedule functionality allows you to organize the shaping queue into different queues. Examples of this functionality are Class Based Weighted Fair Queuing (CBWFQ) and Low Latency Queuing (LLQ).

Note: Traffic shaping and traffic policing both enforce a configured traffic rate. Shaping is an egress queuing mechanism. Policing can be applied inbound or outbound, depending on platform and feature support.

Selection Criteria

The next table lists the differences between shaping and policing to help you choose the appropriate traffic solution.

	Traffic Shaping	Traffic Policing
Primary Behavior	Buffer and queue excess packets over the	Drops (or remarks) excess packets over

	committed rates.	the committed rates. Does not buffer.*
Token Refresh Rate	Tokens are replenished periodically at each **Tc** interval. At the start of each interval, the shaper adds **Bc** tokens, up to the configured bucket limit. **Tc** is constrained by platform-specific limits.	Tokens are replenished based on elapsed time between packet arrivals. The policer calculates the number of tokens to add with: ((t - t1) * policer_rate_bps) / 8, where t - t1 is the time since the previous packet.
Token Values	Configured in bits per second (bps).	Configured in bytes.
Configuration Options	• **Shape** command in MQC to implement class-based shaping. • **Frame-relay traffic-shape** command to implement FRTS. • **Traffic-shape** command to implement GTS.	• **Police** command in the MQC to implement class-based policing. • **Rate-limit** command to implement CAR.
Typical Direction	Outbound	Inbound or outbound, platform-dependent
Bursts	Controls bursts and smooths the output rate over at least eight time intervals. It uses token bucket metering and queues excess packets. The queued packets are released over time, which creates a smoothing effect similar to a leaky bucket.	Allows bursts within configured burst limits. It does not buffer packets to smooth the transmission rate.
Advantages	Less likely to drop excess packets since excess packets are buffered. (Buffers packets up to the length of the queue. Drops can occur if excess traffic is sustained at high rates.) Typically avoids retransmissions due to dropped packets.	Controls the output rate through packet drops. Avoids delays due to queuing.
Disadvantages	Can introduce delay due to queuing, particularly deep queues.	Drops excess packets (when configured), throttled TCP window sizes, and reduces the overall output rate of affected traffic streams. Overly aggressive burst sizes can lead to excess packet drops and throttle the overall output rate, particularly with TCP-based flows.
Optional Packet Remarking	No	Yes, it can optionally transmit, drop, or mark packets, including IP precedence or DSCP actions, depending on the configured MQC policer actions and platform support.



Note: Although policing does not apply a buffer, a configured queuing mechanism applies to conformed packets that need to be queued while they wait to be serialized at the physical interface.

Token Refresh Rate

A key difference between shaping and policing is the rate at which tokens are replenished. Both shaping and policing use the token bucket metaphor. A token bucket itself has no discard or priority policy.

With token bucket functionality:

- Tokens are put into the bucket at a certain rate.
- Each token gives permission for the source to send a certain number of bits into the network.
- To send a packet, the traffic regulator must be able to remove from the bucket a number of tokens equal in representation to the packet size.
- If not enough tokens are in the bucket to send a packet, the packet either waits until the bucket has enough tokens (in the case of a shaper), or the packet is discarded or marked down (in the case of a policer).
- The bucket itself has a specified capacity. If the bucket fills to capacity, new tokens that arrive are discarded and are not available to future packets. Thus, at any time, the largest burst a source can send into the network is roughly proportional to the size of the bucket. A token bucket permits burstiness but bounds it.

Shaping increments the token bucket at timed intervals that use a bits per second (bps) value. A shaper uses the next formula:

<#root>

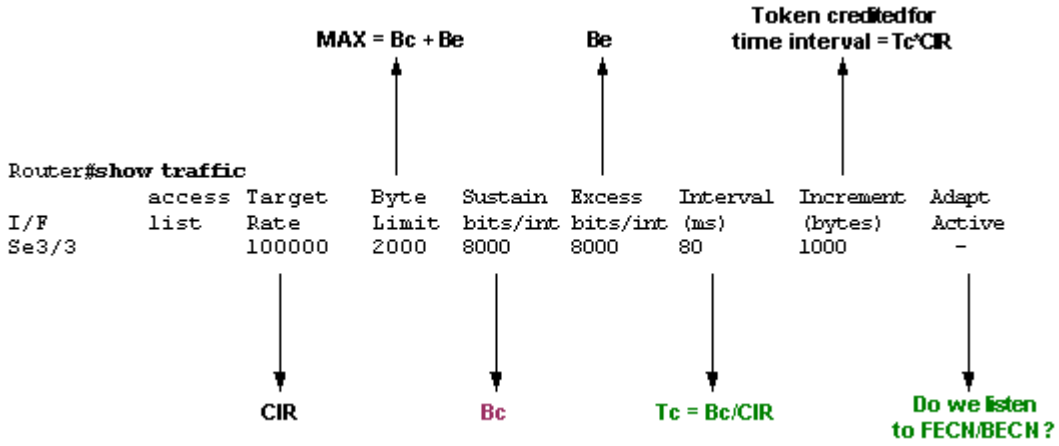
Tc = Bc/CIR

(in seconds)

In this equation:

- **Bc**: Committed burst. The amount of traffic allowed to be sent during one timing interval.
- **CIR**: Committed information rate. The average traffic rate that the shaper or policer enforces.
- **Tc**: Committed time interval. The time interval used by a shaper to send the committed burst while maintaining the configured CIR in seconds.

The range for Tc is between 10 ms and 125 ms. In some legacy platforms with Distributed Traffic Shaping (DTS), the minimum Tc is 4 ms. The router internally calculates this value based on the CIR and Bc values. If Bc/CIR is less than 125 ms, it uses the Tc calculated from that equation. If Bc/CIR is more than or equal to 125 ms, it uses an internal Tc value if Cisco IOS determines that traffic flow can be more stable with a smaller interval. Use the **show traffic-shape** command to determine whether your router uses an internal value for Tc or the value that you configured at the command-line.



show traffic output

When the Excess Burst (Be) is configured to a value different than 0, the shaper allows tokens to be stored in the bucket, up to Bc + Be. The largest value that the token bucket can ever reach is Bc + Be, and overflow tokens are dropped. The only way to have more than Bc tokens in the bucket is to not use all Bc tokens during one or more Tc. Since the token bucket is replenished every Tc with Bc tokens, you can accumulate unused tokens for later use up to Bc + Be.

In contrast, class-based policing and rate-limiting replenish tokens based on the elapsed time between packet arrivals. When a packet arrives, the policer calculates the number of tokens to add from the time since the previous packet and the configured policer rate. Specifically, the token arrival rate is calculated as follows:

```
Tokens added = ((t - t1) * policer_rate_bps) / 8 bytes
```

In other words, if the previous arrival of the packet was at t1 and the current time is t, the bucket is updated with t-t1 worth of bytes based on the token arrival rate.

Note: A traffic policer uses burst values specified in bytes, and the previous formula converts from bits to bytes.

This is an example that uses a CIR (or policer rate) of 8000 bps and a normal burst of 1000 bytes:

<#root>

```
Router(config)#
policy-map police-setting
Router(config-pmap)#
class access-match
Router(config-pmap-c)#
police 8000 1000 conform-action transmit exceed-action drop
```

The token buckets start full at 1000 bytes. If a 450 byte packet arrives, the packet conforms because enough bytes are available in the token bucket. The conform action (transmit) is taken by the packet and 450 bytes are removed from the token bucket (and leave 550 bytes). If the next packet arrives 0.25 seconds later, 250 bytes are added to the token bucket as per the next formula:

```
(0.25 * 8000)/8
```

1. Bucket starts full:
    - Normal burst =**1000 bytes**
    - Initial tokens =**1000 bytes**
2. First packet arrives:
    - Packet size =**450 bytes**
    - The packet conforms because the bucket has enough tokens.
    - The policer removes**450 bytes worth of tokens**from the bucket.

    Calculation: **1000 - 450 = 550 bytes remaining**
3. Next packet arrives 0.25 seconds later:
    - Policer rate =**8000 bps**
    - Tokens added during 0.25 seconds:

    (0.25 × 8000) / 8 = 250 bytes
4. Bucket after refresh:
    - Existing tokens remaining =**550 bytes**
    - New tokens added =**250 bytes**

    Calculation: **550 + 250 = 800 bytes**

The calculation leaves 800 bytes in the token bucket (**tokens remaining in the policer bucket**). If the next

packet is 900 bytes, the packet exceeds and the exceed action (drop) is taken. No bytes are taken from the token bucket.

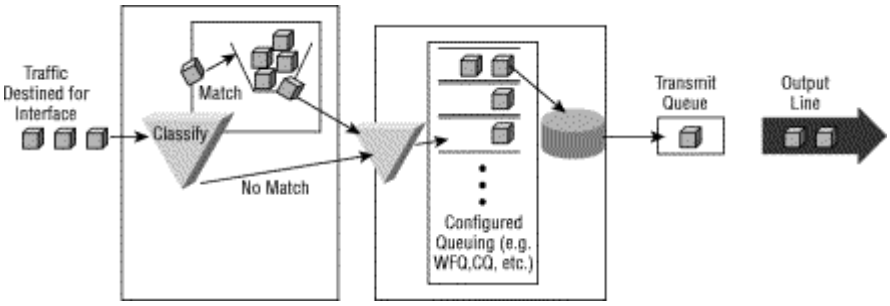
Traffic Shaping

Cisco IOS® supports the next methods of traffic shaping:

- Generic Traffic Shaping
- Frame Relay Traffic Shaping
- Class-Based Shaping and Distributed Class-Based Shaping

All traffic shaping methods are similar in implementation, though their command-line interfaces (CLIs) differ somewhat, and they use different types of queues to contain and shape traffic that is deferred. Cisco recommends class-based shaping and distributed shaping, which are configured with the modular QoS CLI.

The next diagram illustrates how a QoS policy arranges traffic into classes, and queues packets that exceed the configured shaping rates.



Traffic Policing

Cisco IOS supports the next methods of traffic policing:

- Committed Access Rate
- Class-Based Policing

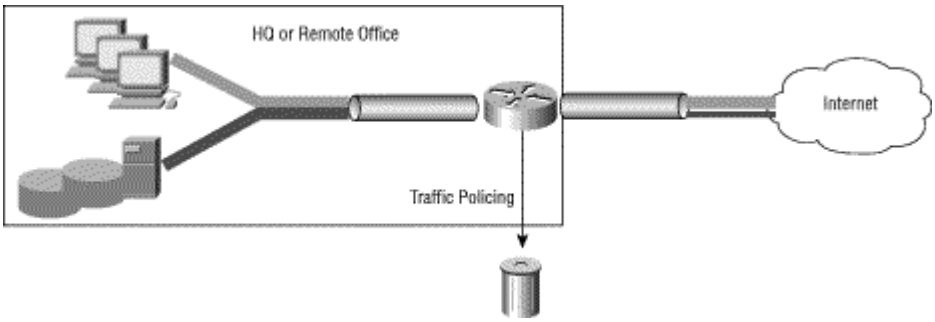
The two mechanisms have important functional differences, as explained in How to Configure Class-Based Policing. Cisco recommends class based policing and other features of the modular QoS CLI when QoS policies are applied.

Use the **police** command to specify that a class of traffic must have a maximum rate imposed on it, and if that rate is exceeded, an immediate action must be taken. In other words, with the **police** command, it is not

an option to buffer the packet and later send it out, as is the case for the **shape** command.

In addition, with policing, the token bucket determines whether a packet exceeds or conforms to the applied rate. In either case, policing implements a configurable action, which includes the IP precedence or Differentiated Services Code Point (DSCP).

The next diagram illustrates a common application of traffic policing at a congestion point, where QoS features generally apply.



Minimum Versus Maximum Bandwidth Controls

The **shape** and **police** commands limit traffic to a configured maximum rate. In MQC, the configured rate for shape average and police is specified in bps unless the platform specific syntax states otherwise. Importantly, neither mechanism provides a minimum bandwidth guarantee during periods of congestion. Use the **bandwidth** or **priority** command to provide such guarantees.

A hierarchical policy uses two service policies – a parent policy to apply a QoS mechanism to a traffic aggregate, and a child policy to apply a QoS mechanism to a flow or subset of the aggregate. Logical interfaces, such as sub-interfaces and tunnel interfaces, require a hierarchical policy with the traffic-limiting feature at the parent level and queuing at lower levels. The traffic-limiting feature reduces the output rate and (presumably) creates congestion, as seen by queuing excess packets.

The next configuration is sub-optimal and is shown to illustrate the difference between the **police** versus the **shape** command when limiting a traffic aggregate, in this case class-default, to a maximum rate. In this configuration, the **police** command sends packets from the child classes based on the size of the packet and the number of bytes that remain in the conform and exceed token buckets. (See Traffic Policing.) The result is that rates given to the Voice over IP (VoIP) and Internet Protocol (IP) classes cannot be guaranteed since the police feature overrides the guarantees made by the priority feature.

However, if the **shape** command is used, the result is a hierarchical queuing system, and all guarantees are made. In other words, when the offered load exceeds the shape rate, the VoIP and IP classes are guaranteed their rate, and the class-default traffic (at the child level) incurs any drops.

⚠ **Caution**: This configuration is not recommended, and is shown to illustrate the difference

⚠ between **police** versus **shape** command when it limits a traffic aggregate.

<#root>

```
class-map match-all IP
   match ip precedence 3
class-map match-all VoIP
   match ip precedence 5
!
policy-map child
  class VoIP
     priority 128
   class IP
     priority 1000
!
 policy-map parent
   class class-default
     police 3300000 103000 103000 conform-action transmit exceed-action drop
     service-policy child
```

In order for the previous configuration to make sense, the policing must be replaced by shaping. For example:

<#root>

```
policy-map parent
    class class-default
     shape average 3300000 103000 0
     service-policy child
```

Use **show policy-map interface <interface>** to verify configured policies, class counters, offered rate, drop counters, shaping rate, queue depth, and policer conform/exceed/violate counters.

<#root>

Router#

show policy-map interface gigabitEthernet 0/0/2

```
 GigabitEthernet0/0/2

 Service-policy output: parent

    Class-map: class-default (match-any)
```

```
      0 packets, 0 bytes
      5 minute offered rate 0000 bps, drop rate 0000 bps
      Match: any
      Queueing
      queue limit 64 packets
      (queue depth/total drops/no-buffer drops) 0/0/0
      (pkts output/bytes output) 0/0

shape (average) cir 3300000, bc 103000, be 0 target shape rate 3300000


Service-policy : child

        queue stats for all priority classes:
          Queueing
          queue limit 512 packets
          (queue depth/total drops/no-buffer drops) 0/0/0
          (pkts output/bytes output) 0/0

        Class-map: VoIP (match-all)
          0 packets, 0 bytes
          5 minute offered rate 0000 bps, drop rate 0000 bps
          Match: ip precedence 5

Priority: 128 kbps, burst bytes 3200, b/w exceed drops: 0



        Class-map: IP (match-all)
          0 packets, 0 bytes
          5 minute offered rate 0000 bps, drop rate 0000 bps
          Match: ip precedence 3

Priority: 1000 kbps, burst bytes 25000, b/w exceed drops: 0



        Class-map: class-default (match-any)
          0 packets, 0 bytes
          5 minute offered rate 0000 bps, drop rate 0000 bps
          Match: any

          queue limit 64 packets
          (queue depth/total drops/no-buffer drops) 0/0/0
          (pkts output/bytes output) 0/0
```



Note: To learn more about parent and child policies, refer toQoS: Child Service Policy for Priority Class.

The previous example is intentionally suboptimal and is provided only to illustrate the difference between policing and shaping in a hierarchical QoS policy. In a recommended QoS design, the **priority** command is typically reserved for real-time, latency-sensitive traffic such as voice. Non-real-time classes that require a minimum bandwidth guarantee commonly use the **bandwidth** command with CBWFQ.

This approach aligns with Cisco QoS design guidance for LLQ and CBWFQ. LLQ provides strict priority treatment for delay-sensitive traffic, while CBWFQ provides bandwidth guarantees for other traffic classes during congestion. Therefore, the IP class in this example is better represented with bandwidth instead of priority, unless that class carries traffic that explicitly requires strict priority treatment.

```
policy-map child
 class VoIP
  priority 128
 class IP
  bandwidth 1000
```

Related Information

- Quality of Services (QoS) Technology Support
- Technical Support & Documentation - Cisco Systems